BZOJ 部分题目解题报告

By: wyx528

1002: [FJOI2007]轮状病毒

打表前几项，发现规律就好了。前几项依次是 1，5，16，45，121，320，841，2205。我们观察奇数项，分别是 1^2，4^2，11^2，29^2。而 1 和 4 我们先不考虑，发现从第三个开始，总是等于上一个的三倍，再减去上上一个。比如 11＝3×4－1，29＝3×11－4。而对于所有的偶数项，只要都除以 5，便也可以得到同样的规律。最后用高精度算一下就可以了。

1003: [ZJOI2006]物流运输 trans

这道题的算法主要是最短路和 DP。我们设图中有 n 个节点，我们一共需要走 D 天。我们首先根据输入信息预处理出 $cost[i][j]$，表示第 i 天到第 j 天走相同的路线，从点 1 到达点 n 需要的最小花费。这个过程可以跑 D^2 次最短路。接着，我们令 $dp[i]$表示第 i 天走完，所需要的最小花费。则有 $dp[i]=\min(cost[1][i]\times i, \min(dp[j]+cost[j+1][i]\times(j-i))$。那么 $dp[D]$就是我们所求的答案。

1004: [HNOI2008]Cards

设题目一共给出了 m 种置换，我们再加上一个$\begin{pmatrix}1,2,...,n\\1,2,...,n\end{pmatrix}$，我们对这 $m+1$ 种情况分别计算答案，由 *Burnside* 引理知，最终的结果为$\frac{\sum_{i=1}^{m+1} f(i)}{m+1}$，其中 $f(i)$表示第 i 种置换的不动点个数。我们设第 i 种置换共有 T 个循环，则对每个循环进行 DP 即可，设 $dp[i][j][k]$表示前 i 个循环，第一种颜色用了 j 个，第二种颜色用了 k 个的方案数，并开一个变量记录前 i 个循环一共有多少张卡片，转移就很容易了，不多说了。最后就有 $f(i)=dp[T][x][y]$，其中 x 为第一种颜色的总张数，y 为第二种颜色的总张数。

1005: [HNOI2008]明明的烦恼

有一种东西叫做树的 *prufer* 编码，一个树如果确定，那么它的 *prufer* 编码也确定，如果确定了一个 *prufer* 编码，那么这个树也确定。一个由 n 个结点组成的树中，*prufer* 编码的长度为 $n-2$，若点 i 的度为 $d[i]$，则 i 会在 *prufer* 编码中出现 $d[i]-1$ 次。我们设输入数据中确定度的点有 m 个，那么不确定就有 $n-m$ 个，这确定的 m 个的点在 *prufer* 编码中出现的次数也将确定，我们记 $S=\sum_{i=1}^{m}(d[i]-1)$，那么我们首先把这 S 个数组成的排列，共有 $\frac{S!}{\prod_{i=1}^{m}(d[i]-1)!}$ 种方案，考虑我们选择的 S 个位置，有 $C(n-2,S)$ 种选法，对于度自由的 $n-m$ 个点，共有

$(n-m)^{n-2-S}$ 种放法，所以答案就是 $\frac{S!}{\prod_{i=1}^{m}(d[i]-1)!}\times C(n-2,S)\times(n-m)^{n-2-S}$。

1007：[HNOI2008]水平可见直线

我们用一个栈来维护直线间的关系。首先我们按照直线的斜率从小到大进行排序，斜率一样的按照截距从大到小进行排序，那么我们可以发现对于斜率相同的一组直线，只有截距最大的那个是有可能被看到的，所以我们先删掉这些不可能的直线。对于当前的直线，我们将它与栈顶直线的交点设为 A，与栈顶的下一条直线的交点设为 B。我们发现如果 A 的横坐标大于等于 B 的横坐标，那么栈顶直线必被遮盖，所以弹出直到栈顶直线不被遮盖后，把当前直线入栈，那么最后栈中的所有直线都是可见直线。

1009：[HNOI2008]GT 考试

由于只有一个串，用 KMP 就可以了，多个串的话建立 AC 自动机。求出这样的状态：对于 AC 自动机的第 i 个节点，如果再扩展出 0~9 中的一个数字，将转移到哪个节点，然后用矩阵乘法推一遍就可以了。

1010：[HNOI2008]玩具装箱 toy

具有决策单调性的 DP，用四边形不等式优化到 $O(n\log n)$，或者直接斜率优化，时间复杂度 $O(n)$。

1011：[HNOI2008]遥远的行星

我们设 i 可以受到的引力范围为[1,$g[i]$]，$f[i]$表示 i 受到的引力总和，则 $f[i]=\sum_{j=1}^{g[i]}\frac{m[i]\times m[j]}{i-j}$，化简一下有 $f[i]=m[i]\sum_{j=1}^{g[i]}\frac{m[j]}{i-j}$，然后考虑间隔为 T 的 $f[i+T]$，$f[i+T]=m[i+T]\sum_{j=1}^{g[i+T]}\frac{m[j]}{i+T-j}$，我们对 $f[i+T]$ 进行一下整理：

$f[i+T]=m[i+T]\left(\sum_{j=1}^{g[i]}\frac{m[j]}{i+T-j}+\sum_{j=g[i]+1}^{g[i+T]}\frac{m[j]}{i+T-j}\right)$，继续整理：$f[i+T]=m[i+T]\left(\sum_{j=1}^{g[i]}\frac{m[j]}{i-j}\times\frac{i-j}{i+T-j}+\sum_{j=g[i]+1}^{g[i+T]}\frac{m[j]}{i+T-j}\right)$，

估算一下，$f[i+T]\approx m[i+T]\left(\sum_{j=1}^{g[i]}\frac{m[j]}{i-j}\times\frac{i-\frac{g[i]}{2}}{i+T-\frac{g[i]}{2}}+\sum_{j=g[i]+1}^{g[i+T]}\frac{m[j]}{i+T-j}\right)=m[i+T]\left(\frac{f[i]}{m[i]}\times\frac{i-\frac{g[i]}{2}}{i+T-\frac{g[i]}{2}}+\sum_{j=g[i]+1}^{g[i+T]}\frac{m[j]}{i+T-j}\right)$，

其中 T 我们可以取一个常量，这样时间复杂度就位 $O(nT)$，n 最大是 10^5，T 在这里取 800 就可以通过所有测试数据。

1013：[JSOI2008]球形空间产生器 sphere

这道题主要是构造出方程组，然后用高斯消元。我们设输入矩阵的第 i 行第 j 列的元素值为 $a_{i,j}$，我们所求的

坐标为 x_i。则：$\forall 2 \le k \le n+1$，有 $\sum_{i=1}^{n}\left(x_i - a_{1,i}\right)^2 = \sum_{i=1}^{n}\left(x_i - a_{k,i}\right)^2$，整理得 $\sum_{i=1}^{n} 2x_i \times \left(a_{k,i} - a_{1,i}\right) = \sum_{i=1}^{n}\left(a_{k,i}^2 - a_{1,i}^2\right)$。接下来使用高斯消元求解就可以了。

1014：[JSOI2008]火星人 prefix

这道题用一棵 SPLAY 来维护，对于查询操作，可以进行二分答案，每个节点存储当前节点所在那段字符串的一个 HASH 值就可以了。

1015：[JSOI2008]星球大战 starwar

正着做很麻烦，我们倒着考虑这个问题。首先求出 *m* 次删除以后的连通块的个数。然后对删除操作进行倒序处理，如果当前删的这个点不是第一次删除，那么不考虑，否则联通块加一，然后对与这个点相邻的点进行考虑，有可能还涉及到连通块的减一，用并查集维护就可以了。

1016：[JSOI2008]最小生成树计数

这道题首先需要两个引理，第一个是对于一个图的任意一个 MST，相同边权的边的数量相同，第二个是对于一个图的任意一个 MST，当第 *i* 条边处理完成后，并且这条边是相同边权的最后一条，那么图中的连通分量个数相同。而题目中说相同的权值不会超过 10 个，所以我们可以花 2^{10} 的时间去搜索，然后判断连通分量是否和最初的 MST 相同，最后利用乘法原理统计答案就看可以了。

1019：[SHOI2008]汉诺塔

设 *dp*[*i*]表示 *i* 个盘子需要的步数，由于 *dp*[*i*]满足线性关系，所以必有 *dp*[*i*]＝*A*×*dp*[*i*－1]＋*B*。按照题目给出的规则暴力模拟出 *dp*[1]，*dp*[2]，*dp*[3]，然后解出 *A* 和 *B*，然后 *O*(*n*)递推即可。

1022：[SHOI2008]小约翰的游戏 John

Nim 博弈问题。我们分两种情况，如果全是 1 的话，如果是奇数个则是 Brother 否则是 John。如果不全是 1 的话，记 *p* 为所有石子数量的 *xor* 和，若 *p* 为 0 则是 Brother 否则是 John。

1024：[SCOI2009]生日快乐

爆搜就行了，以长为例，设长为 *a*，分成 *k* 分，那么分点一定是 *ia*/*k*。

1025：[SCOI2009]游戏

我们发现如果把 *n* 个数的排列分解成 *m* 个循环的话，那么最少再变回 1,2，……，*n* 的次数就为每个循环大小的最小公倍数。因此原问题等价于把一个自然数 *n* 分解成一些自然数的和，求最小公倍数有多少种情况。如果最小公倍数为 *T* 的话，则 $T = \prod p_i^{a_i}$，$\sum p_i^{a_i} \le n$。因此我们可以考虑把 *n* 以内的素数分解出来，用背包 DP 即

可。$dp[i][j]$表示前 i 个素数，已经用了的和为 j 的方案数，则 $dp[i][j]=dp[i-1][j]+\sum dp[i-1][j-prm[i]^k]$，边界条件 $dp[0][i]=1(0\le i\le n)$，设共有 top 个素数，则答案为 $dp[top][n]$。

1026：[SCOI2009]windy 数

数据范围是 2×10^9，我们可以每隔 100 万计算一个值，把所有的都预处理出来打到一张表里，然后对于询问，最多只需要计算 10^6 个。

1029：[JSOI2007]建筑抢修

我们首先按照结束时间排序，那么对于当前的建筑。如果能修则修，如果不能且当前所需的时间小于之前修完的最小时间，则用这个和之前的那个换一下，整个过程用堆或者优先队列维护就可以了。

1030：[JSOI2007]文本生成器

这个问题很经典了，首先对于输入的所有单词建立 AC 自动机，我们设共有 S 个节点。对于所求答案，我们可以求它的补集，也就是说共有多少种是不含有任何一个单词的，再求出总方案数，我们设总长度为 n，那么显然总方案数为 26^n。状态 $dp[i][j]$表示到第 i 个字符，处于 AC 自动机的第 j 个节点，共有多少种方案。我们考虑在第 j 个节点的下面在插入一个字符(A~Z)可以到达状态 p，这个 p 可以由 $fail$ 求出。那么我们可以用 $dp[i][j]$去更新 $dp[i+1][p]$，其中满足 j 节点是合法的，且 p 节点是合法的。（这里的合法是指没有走到任何一个单词的末尾）那么总共满足要求的不含有任何一个单词的方案数就是 $T=\Sigma dp[n][i]$，其实节点 i 的状态合法。答案则为 26^n-T。由于总长度很小暴力 DP 就可以了，不需要用矩阵优化。

1036：[ZJOI2008]树的统计 Count

第一次写树链剖分，以前觉得很难，发现其实挺好理解的。这道题只需要写一个求 max 和一个求 sum 的函数就可以了，线段树不需要打标记，比较简单。

1037：[ZJOI2008]生日聚会 Party

这是一个不是很好想的 DP 问题。定义状态 $dp[a][b][c][d]$表示前 a 个人中有 b 个人是男生，其中男生最多比女生多 c 个，女生最多比男生多 d 个的方案数。则我们可以用 $dp[a][b][c][d]$去更新 $dp[a+1][b+1][c+1][\max(d-1),0]$以及 $dp[a+1][b][\max(c-1,0)][d+1]$，其中需要满足 $c+1\leqslant p$，$d+1\leqslant p$，p 是题目中输入的第三个整数，边界条件是 $dp[1][1][1][0]=dp[1][0][0][1]=1$。答案是 $\Sigma dp[n+m][n][i][j]$。

1040：[ZJOI2008]骑士

环套树的 DP，做法是这样的。首先很显然对于每一个连通分量可以单独处理，所以我们只考虑任意一个单独的连通分量。为了方便，我们定义每个骑士 i 讨厌的骑士为 $hate[i]$。建图的时候我们反过来考虑，连一条从 $hate[i]$到 i 的边。那么首先找到以环上的点为根的树中的最大值。然后再把环上的任意一个点作为起点推一遍就可以了。

1041：[HAOI2008]圆上的整点

这是一道数学题，其实我们可以只求出第一象限内的个数，然后把这个结果乘 4 加上坐标轴上的 4 个就是最终答案。由题意知 $x^2+y^2=r^2$，整理得 $y^2=(r-x)\times(r+x)$。我们设 $A=r-x$，$B=r+x$，$\gcd(A,B)=p$。则 $A=np$，$B=mp$，$n<m$，$\gcd(n,m)=1$。联立得 $2r=p(n^2+m^2)$。我们只需要枚举 p，然后枚举相应的 n，并计算 m 是否合法即可。枚举 p 为 $\sqrt{n}$ 次，但是 p 为 $2r$ 的约数的并没有多少，枚举 n 为 $\sqrt{p}$ 次。

1042：[HAOI2008]硬币购物

首先不考虑数量限制，$dp[i][j]$表示点 i 种硬币组成价值为 j 的物品的方案数，则有 $dp[0][0]=0$，$dp[i][j]=dp[i-1][j]+dp[i][j-c[i]]$，最终答案就是 $dp[4][n]$。我们考虑个数限制，则答案应为任何一种都无限制的方案，减去第一种硬币个数超了的方案，减去第二种硬币个数超了的方案，减去第三种硬币个数超了的方案，减去第四种硬币个数超了的方案，加上第一种，第二种硬币个数超了的方案，……，加上第一种，第二种，第三种，第四种硬币个数超了的方案，总之就是个容斥原理。我们考虑如何计算第一种硬币个数超了的方案，设面值为 c，数量为 d，则我们必须要用第一种硬币选出 $(d+1)\times c$ 的价值，然后剩下的 $n-(d+1)\times c$ 随便选，因此第一种硬币个数超了的方案就是 $dp[4][n-(d+1)\times c]$，剩下情况类似，这样每次询问的复杂度就做到了 $O(1)$。

1045：[HAOI2008] 糖果传递

和 2620 一样了，前缀和，排序，中位数，随便搞搞。

1046：[HAOI2007]上升序列

水题，第一问正常求，第二问贪心，从前往后扫，能输出就输出。

1047：[HAOI2007]理想的正方形

如果是一维情形的话那就是经典问题了。考虑二维情形，我们首先求出<*i*,*j*>这个格子往左 n 个的最大值和最小值，求两遍单调队列就可以了。然后我们在用求出的结果再求出格子<*i*,*j*>往上 n 个的最大值的和最小值。这样就求出了以<*i*,*j*>为右下角的矩阵中，最大的数和最小的数分别是多少，问题就解决了。时间复杂度为 $O(ab)$。

1050：[HAOI2006]旅行 comf

由于边权的大小是有界的，不超过 3 万。所以我们可以枚举经过边的最小值，然后用 SPFA 的方法求出经过的最大边的最小值并更新答案即可。

1053：[HAOI2007]反素数 ant

我们按照每个质因子的个数进行搜索，显然质因子最多只能到 29。最多只有 10 个质数，爆搜就可以了。

1054：[HAOI2008]移动玩具

由于棋盘是 4×4 的，所以我们可以把棋盘翻译成一个 16 位的二进制数，那么最多只有 $2^{16}=65536$ 种情况，然后用 BFS 解决，注意判重。

1055：[HAOI2008]玩具取名

定义状态 $dp[i][j][k]$表示[i,j]组成的字符串能否合并成字母 k，初值是 $dp[i][i][a[i]]=true$，转移就很显然了，时间复杂度 $O(n^3)$。

1056：[HAOI2008]排名系统

这道题我们用一棵平衡树来维护分数的关系，每个节点保存三个域，两个整数分别为分数和编号，一个 string 为姓名。这题有一个隐含条件，就是分数一样的话第二关键字是加入系统的顺序，也就是我们要在每个节点设置编号的原因。更恶心的是，如果同一个同学的成绩被修改，那么他的编号也应该做相应的修改，也就是说，如果我们遇到了修改之前加入的同学，我们应该先删掉他原来的所有信息，再把他作为一个全新的同学处理。对于编号和分数的维护过程，我们可以用 C＋＋的 map 来实现，开两个 HASH 就可以了。

1057：[ZJOI2007]棋盘制作

第一问很常规的 DP。对于第二问，我们设第 i 行第 j 列的格子为 $a[i][j]$。则我们首先把所有 $i+j$ 为偶数的格子进行取反（当然奇数也可以）。然后就变成了最大子矩阵问题。我们要分别求全 0 和全 1 的最大子矩阵，跑完一遍以后把整个矩阵取反再跑一遍就可以了。

1058：[ZJOI2007]报表统计

实在懒得写 Splay 和 SBT 了。开两个 STL 的 set 就可以了。记一下每个数被添加一些数以后最左面的是多少，最右面的是多少。为了防止超时，如果 MIN_SORT_GAP 已经为 0，那么就不更新它了。

1060：[ZJOI2007]时态同步

首先吐槽一下，这道题数据的后三个点是错误的，需要打表。然后说做法。做法就是宽搜一遍，得到一个层次遍历顺序。以这个顺序的倒序推一遍，推法是对于当前的一个点 p，设它到根节点的距离为 $dis[p]$，设它的儿子节点为 $s_1,s_2,s_3\cdots\cdots$。我们选出 dis 最大的一个 s_k。然后对于每个儿子 s_i，将答案加上 $dis[s_k]-dis[s_i]$。做完以后把 $dis[p]$修改为 $dis[s_k]$就可以了。

1064：[Noi2008]假面舞会

我们讨论四种情况：

① 一条链，这种显然对答案没有影响，每个人随便分配即可。

② 存在有向环，设环中节点为 p 个，那么答案就被限制成了 p 的约数，如果有多个环，要满足每个环的限制。

③ 存在两条链，这两条链有公共的起点和端点，我们设第一条链有 p 个点，第二条链有 q 个点，不妨设 $p>q$，假如答案为 x，则 x 一定是 $p-q$ 的约数，否则在这两条链上就矛盾了。与②相同，如果存在多个这种两条链的结构要都满足要求。

④ 存在一种有向无环图，我们把有向边弱化成无相边以后，仍然是无环的。这就像一个层次结构，如果这个结构里有 p 个点，共 q 层，那么就限制了这 p 个点最多只能有 q 种不同的帽子，最少则可以有 2 种。

如果我们发现图中存在②或者③，则最大值就为公共公约数，最小值则为公共公约数的最小的不小于 3 的约数，否则的话最大值就是④中每个结构 q 的总和，最小值就是 3。特判一下如果最大值也小于 3 的话，那么输出两个－1。对于如何寻找②和③这种结构，我们把原图的一条有向边 a 到 b 拆成两条无相边，a 到 b 边权为 1，b 到 a 边权为－1。这样对原图进行 *dfs*，如果点 a 的下一个要搜索的点 b 已经被搜索过，那么就相当于是找到了一个环，环上的点的个数就是 abs(*dep*[*a*]－*dep*[*b*])。

1066：[SCOI2007]蜥蜴

简单的网络流。把每一个点(*i*,*j*)拆成(*i*,*j*,0)和(*i*,*j*,1)。如果一个格子的高度 *h*(*i*,*j*)大于 0，则连边(*i*,*j*,0)，(*i*,*j*,1)流量为 *h*(*i*,*j*)。若一个格子有蜥蜴，则连边(*S*,(*i*,*j*,0))。若一个格子可以跳出边界则连边((*i*,*j*,1),*T*)流量为正无穷。若格子(*i*,*j*)可以到(*i*’,*j*’)则连边((*i*,*j*,1),(*i*’,*j*’,0))流量为正无穷。答案就是总数减去最大流。

1067：[SCOI2007]降雨量

由于数据是有序的，所以不需要排序。我们对整个数据建立线段树，对于每个节点维护两个域，第一个是当前序列的最大值，第二个是当前序列是否为连续的，所谓连续是指数据[*L*,*R*]中的任意一个在这里都可以取到。接下来对于询问分为四种情况，我们设读入的两个数分别为 x 和 y：①x 可以找到，y 可以找到，这种情况直接判断就可以了，注意是否连续，以及边界情况。②x 不可以找到，y 不可以找到，这种情况直接输出 maybe。③x 可以找到，y 不可以找到，那么我们把 y 换成 y 的前驱，也就是小于 y 的最大的值。然后判断 x 是否比这一段的任意一个都大，是的话输出 maybe 否则输出 false。④x 不可以找到，y 可以找到，那么我们把 x 换成 x 的后继，也就是大于 x 的最小的值。然后判断 x 到 $y-1$ 这一段中，如果 y 比任意一个大，则输出 maybe，否则输出 false，对于查着过程，以及前驱和后继都用二分实现。

1070：[SCOI2007]修车

建立费用流模型。我们设源点为 S，汇点为 T。S 向每一个车连容量为 1 费用为 0 的边。接着，我们将技术人员复制 n 份，设读入的数组为 *cost*[*i*][*j*]，那么我们由每一个车 i 向第 k 份中的技术人员 j 连容量为 1 费用为 $k\times$*cost*[*i*][*j*]的边，以及第 k 份中的技术人员 j 连向 T 容量为 1 费用为 0 的边。最小费用就是答案。

1071：[SCOI2007]组队

暴力枚举身高，然后扫一遍速度就行了。时间复杂度 $O(n^2)$。

1072：[SCOI2007]排列 perm

简单状压 DP。$dp[i][j]$表示在二进制表示的状态 i 下，余数为 j 的方案数，其中 i 表示每个数是否用过，然后转移就很显然了。最后处理一下重复的数，除一下就可以了。

1076：[SCOI2008]奖励关

由于宝物的种类很小，容易想到状压 DP。$dp[i][j]$表示前 i 次游戏，当前宝物状态是 j 能得到的期望分数。转移很简单不说了，注意正着推比较繁琐，因为可能涉及到用没出现的状态去更新出现的状态，所以倒着推就可以了。

1079：[SCOI2008]着色方案

由于每种颜色都不超过 5 个，我们定义 $dp[a][b][c][d][e][last]$表示能用 1 次的有 a 种，能用 2 次的有 b 种，……，能用 5 次的有 e 种，其中上一次选的是能用 $last$ 次的方案数，然后用记忆化搜索实现。

1082：[SCOI2005]栅栏

首先二分答案，对于当前等待检验的 mid，我们要切出 mid 块，一定是去切最短的 mid 块，然后搜索就可以了。

1084：[SCOI2005]最大子矩阵

一行的太简单了，不说了。对于两行的，定义状态 $dp[k][i][j]$表示选了 k 个矩形，对于第 1 列中，最后一个方格是第 i 个，对于第 2 列，最后一个方格是第 j 列，那么显然我们可以用 $dp[k-1][i'][j]$和 $dp[k-1][i][j']$来更新 $dp[k][i][j]$，特别要注意的是当 $i=j$ 时，还需要用 $dp[k-1][i'][j']$来更新 $dp[k][i][j]$，时间复杂度是 $O(n^3m)$，n 是列数，m 是选择的矩阵个数。

1085：[SCOI2005]骑士精神

迭代加深搜索就可以过了，剪枝统计一下当前有多少个不归位的，如果这个个数加上已经搜过的深度仍然大于深度上限则剪枝。

1087：[SCOI2005]互不侵犯 King

预处理相邻两行最多 512×512 种情况是否可行，然后状压 DP 就可以了。

1088：[SCOI2005]扫雷 Mine

我们枚举第一个格子为 0 或者 1，则剩下的格子可以根据第一个格子推算。所以最终的答案只能是 0 或 1 或 2。

1089：[SCOI2003]严格 n 元树

记 $dp[i]$表示深度为 i 的答案，$s[i]=\sum_{j=0}^{i}dp[i]$，则有 $dp[i]=s[i-1]^n+s[i-2]^n$，推倒过程显然，需要写高精度的加减乘，边界条件 $dp[0]=dp[1]=1$。

1090：[SCOI2003]字符串折叠

区间 DP，状态 $dp[i][j]$表示 i 到 j 的字符串折叠完成的最小长度，第一种方式就是将 i 到 k 和 $k+1$ 到 j 两段直接加起来，所以 $dp[i][j]=\min(dp[i][k]+dp[k+1][j])$，$i<=k<j$，第二种方式是进行折叠，$dp[i][j]=\min(calc(i,j,k))$，其中当 i 到 j 的长度不能被 k 整除或者 i 到 j 无法拆成每段长度都是 k 的一样的字符串时，$calc(i,j,k)=+\infty$，否则 $calc(i,j,k)=getw(j-i+1)+2+dp[i][i+k-1]$，$getw(x)$是返回 x 的十进制有多少位。答案是 $dp[1][n]$，时间复杂度 $O(n^3)$。

1095：[ZJOI2007]Hide 捉迷藏

见曹钦翔论文《数据结构的提炼与压缩》。用括号序列算一些东西，然后扔到线段树里就行了。

1096：[ZJOI2007]仓库建设

DP，需要斜率优化。我们先写出朴素的 DP 方程：$dp[i]$表示第 i 个地点必须建立仓库所需要的最小花费，则有 $dp[i]=\min(dp[j]+w[j,i])+c[i]$。其中 $w[j,i]=p[j+1]\times(x[i]-x[j+1])+p[j+2]\times(x[i]-x[j+2])+\cdots\cdots+p[i]\times(x[i]-x[i])$，$0<=j<i$。我们对 $w[j,i]$进行化简，则 $w[j,i]=x[i]\times(p[j+1]+p[j+1]+\cdots\cdots+p[i])-(p[j+1]\times x[j+1]+p[j+2]\times x[j+2]+\cdots\cdots+p[i]\times x[i])$。为了简化问题和后面的描述，我们设 $sump[i]=p[1]+p[2]+\cdots\cdots+p[i]$，$sum[i]=p[1]\times x[1]+p[2]\times x[2]+\cdots\cdots+p[i]\times x[i]$。则 $w[j,i]=x[i]\times(sump[i]-sump[j])-(sum[i]-sum[j]$。带入原方程并整理得到 $dp[i]=\min(dp[j]-x[i]\times sump[j]+sum[j])+R[i]$。其中 $R[i]=x[i]\times sump[i]-sum[i]+c[i]$，$R[i]$是一个与决策无关并且可以预处理得到的常数。这个方程具有决策单调性（证明略，其实我也不会证）。我们考虑 min 中的内容。设 $a[i]=-x[i]$，$x[j]=sump[j]$，$y[j]=dp[j]+sum[j]$。则 $dp[i]=\min(a[i]\times x[j]+y[j])+R[i]$。我们令 $T=a[i]\times x[j]+y[j]$，则 $y[j]=-a[i]\times x[j]+T$。我我们所需要的正是这个最小的 T。由题意只 $-a[i]$，$x[i]$，$y[i]$均随着 i 的增加而增加。对应地，我们把 $x[i]$，$y[i]$放入平面直角坐标系中，显然参与决策的点在一个下凸的凸包上。维护一个双端队列，对于第 i 个决策，我们从队首开始扫描，把不好的决策都删掉以后，我们用新的队首作为 i 的决策，求出 $dp[i]$，并计算出 $x[i]$，$y[i]$。接着我们用 Graham 维护凸包的方式，删掉队尾不在新凸包上的点，并把 $x[i]$，$y[i]$加进新凸包。每个元素最多进队一次，出队一次，被扫描一次，整个算法时间复杂度为 $O(n)$。

1097：[POI2007]旅游景点 atr

由于访问的景点数非常少，设共有 p 个，可以考虑状压 DP 解决。$dp[i][j]$表示在点 i，已经访问的点用二进制表示为 j，所需的最短时间。则我们可以用 $dp[i][j]+dist[i][k]$去更新 $dp[k][j+2^k]$，能更新的前提是 j 中的景点包含了 k 的所有前驱景点。上面的那个 $dist[i][k]$可以通过求 p 次的单源最短路得到。最后我们的答案就是 $\min(dp[i][2^p-1]+dist[i][n])$，时间复杂度 $O(pm\log n+2^pp^2)$。而 POI 官网上只给了 64MB 的内存，如果开 20×2^{20} 的数组就会 MLE，一个解决办法就是我们观察到对于 $dp[i][j]$，j 的二进制第 i 位必须为 1，否则是不合法的，因此 p 位二进制有用的只有 $p-1$ 位，我们可以考虑压缩掉一位，这样空间就变成了 20×2^{19}，不会 MLE 了。

1098：[POI2007]办公楼 biu

最直接的想法就是构造这个图的反图如去求连通块的个数及每个连通块的大小。但是构建反图需要 $O(n^2)$的时间，会超时。我们不妨这样想，n 个点 m 条边的无向图，记第 i 个连与 $d[i]$个点连通，则最小的 $d[i]$一定不会超过$\left\lfloor \frac{2m}{n} \right\rfloor$，这个很显然。我们记 $d[i]$最小的点为点 p，则与点 p 不相邻的 $n-d[i]$个点一定都要跟点 p 放在一起，这样我们就把这 $n-d[i]$个点合并了，对于剩下的 $d[i]$个点只要暴力求就可以了。

1100：[POI2007]对称轴 osi

考虑求一个多边形有多少条对称轴，我们可以把多边形表示成一个由边长和角度组成的序列，这样如果两个点的连线 i 和 j 是一条对称轴的话，那么这两个点分割成的两个序列连起来应该是一个回文串。具体方法就是我们找一个长度为 $2n$ 的序列，第一个是边长，第二个是角度，第三个是边长，依次类推。记得到的这个串为 S，我们复制 S 得到 SS，及 S 的反串为 S'，则原问题就变成了 S'在 SS 中出现了多少次，这个可以用 kmp 解决。注意如果角度会挂精度的话可以直接用叉积来代替，因为如果边长不等的话肯定就不行了，如果边长相等的话叉积不等，就说明了角度不等。

1101：[POI2007]Zap

原题所求为$\sum_{i=1}^{n}\sum_{j=1}^{m}1\left(\gcd(i,j)=d\right)$，我们设 $N=\left\lfloor \frac{n}{d} \right\rfloor$，$M=\left\lfloor \frac{m}{d} \right\rfloor$，

则原式等于$\sum_{i=1}^{N}\sum_{j=1}^{M}1\left(\gcd(i,j)=1\right)$。

根据莫比乌斯反演，$\sum_{d|n}m(d)=\begin{cases}1 & n=1\\ 0 & n\neq 1\end{cases}$，

因此原式等于$\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{d|\gcd(i,j)}m(d)$，

继续化简得$\sum m(d)\sum_{1\le i\le N\text{且}d|i}\sum_{1\le j\le M\text{且}d|j}1$，

把右面的两个求和合并一下可以得到$\sum m(d)\left\lfloor \frac{N}{d} \right\rfloor\left\lfloor \frac{M}{d} \right\rfloor$，

由于$\left\lfloor \frac{N}{d} \right\rfloor$的取值是$\sqrt{N}$ 数量级的，因此可以采用分段的方式在$O(50000)$ 预处理，$O\left(\sqrt{N}+\sqrt{M}\right)$的时间内完成一次询问来解决此题。

1102：[POI2007]山峰和山谷 Grz

直接 floodfill 一次就行，注意要用 bfs 不要用 dfs，会爆栈。

1103：[POI2007]大都市 meg

首先搞出 dfs 序，怕爆栈的话可以人工堆栈，这样把一条边边权变成 0 其实就是把这条边深度较大的那个点

为根节点的子树的答案都减一，由于涉及的询问是单点查询，所以用线段树或者树状数组都可以。时间复杂度 $O((n+m)\log n)$。

1104：[POI2007]洪水 pow

我们首先考虑如果在格子 a 修建一个抽水机，在什么情况下格子 b 的水也可以被抽干。我们可以发现当且仅当存在一条从 a 到 b 的路径，中间经过的抽水机（包括 a）的高度都不大于 b 的高度。因此我们可以考虑把所有格子的高度从小到大排序，我们把每一个格子建成一个集合。然后按照海拔高度从小到大扫描格子，对于当前的格子 i，我们找到所有与 i 相邻并且海拔高度不大于格子 i 的格子，我们发现如果这些格子中的任意一个洪水要是被解决了，那么格子 i 的洪水也可以被解决，所以我们合并这些格子。对于当前的格子 i，如果它必须被清理且与它相邻的格子集合中没有任何一个被清理，我们则把这个集合的清理状态标记为真，然后答案加 1。集合和每个格子是否被清理用并查集来维护就可以了。

1105：[POI2007]石头花园 SKA

神构造题，首先考虑周长最小。将所有的点都变成 $x<=y$ 可以使得第一问最小，证明我也不知道。设这样得到的矩形的左下角为($minx$,$miny$)，右上角为($maxx$,$maxy$)，则第二问只能在下面的四个矩形中选择：

($minx$,$miny$)，($maxx$,$maxy$)

($miny$,$minx$)，($maxx$,$maxy$)

($minx$,$miny$)，($maxy$,$maxx$)

($miny$,$minx$)，($maxy$,$maxx$)

我们逐一检查上面的四种情况，注意一个特殊的地方，如果一个点不在矩形内，不能直接把答案加上代价，要看是否合法，如果这个点不在矩形内，且交换坐标以后仍然不在矩形内，则这个大矩形是不合法的，直接退出即可。时间复杂度 $O(n)$。

1106：[POI2007]立方体大作战 tet

每个数都出现两次，我们记第 i 个数出现的位置分别为 $L[i]$和 $R[i]$，我们考虑两个数 i 和 j，可以发现如果区间[$L[i]$,$R[i]$]和区间[$L[j]$,$R[j]$]交集为空的话，那么无论先消掉谁，结果都一样，如果这两个区间是相交关系，那么也是先消掉谁，结果都一样。对结果有影响的唯一一种可能就是一个区间包含另一个区间，我们不妨假设区间 A 包含了区间 B，那么先消掉 B 再消掉 A 的代价就要比先消掉 A 再消掉 B 的代价少 2。由此我们便得到了一种方法，我们按照每个区间的右端点从小到大排序，然后按照这个顺序依次消去每个编号的方块，就保证了如果 A 包含 B，则先消去的一定是 B 而不是 A，至于统计答案可以用树状数组或线段树维护一下就可以了。

1107：[POI2007]驾驶考试 egz

我们定义 $dpl[i]$表示使得从左数的第 i 条道路能从最下面开始往左走，到达第 1 条道路的最上方需要添加最少多少条道路，$dpr[i]$表示使得从左数的第 i 条道路能从最下面开始往右走，到达第 n 条道路的最上方需要添加最少多少条道路。则我们要求的就是找到一对 L 和 R，使得 $dpr[L]+dpl[R]<=lim$（lim 表示最多可以添加的道路条数），然后最大化 $R-L+1$ 减去在[L,R]内之前就能作为起点的道路数。由于 $dpl[i]$和 $dpr[i]$形式类似，我们只考虑 $dpl[i]$的求解过程。

易知 $dpl[1]=0$，而只由 $dpl[i-1]$转移到 $dpl[i]$比较困难，因此我们考虑把状态升到两维。定义状态 $dp[i][j]$表示位于第 i 条道路的 j 高度，能向左走到达第 1 条道路的最上面所需添加的最少道路数，其中 $1<=i<=n$，$0<=j<=m$。则有：

①．$dp[i][j]=dp[i-1][j]+1$，这种方式表示直接在 $i-1$ 和 i 的 j 位置修建一条道路；

②．$dp[i][j]=dp[i-1][k]$，其中 $k>=j$ 且第 i 条道路的 k 高度已经连接到了第 $i-1$ 条道路，这样我们就可以向上走一段然后再向左，是不需要添加道路的。

而 $dp[i][j]$的值则为上面两种情况的最小值。但是这样转移时间复杂度为 $O(nm)$，无法承受。我们可以发现对于所有的 i，在第二种方式中转移得到的 k 并不多，因此我们可以考虑先全部从第①种方式转移，根据 $dp[i][j]=dp[i-1][j]+1$，既然把每个 j 都加 1，我们就可以开一个全局变量记一下加了多少而不用真的去把每个都加 1。然后我们考虑第②中方式，易知一个 k 的作用范围是$[0,k]$，因此对于一个 k，我们是在做这样一个操作：用 $dp[i-1][k]$来替换所有 $dp[i][j]$中比 $dp[i-1][k]$大的数（$0<=j<=k$），这可以用一棵线段树完成。

实现的时候也有一些技巧，我们把$[0,m]$建到线段树里，对于一个 i（$1<=i<=n$），显然第①种转移方式中需要累加的量为 $i-1$，我们考虑按照 k 从小到大的顺序进行更新，对于一个 k，我们记线段树中节点 k 的值为 $a[k]$，则我们用 $a[k]-1$ 来更新 $a[0..k]$ （至于为什么要减 1，我们可以考虑我们已经把$[0,m]$都按照①方式转移来了，也就是 $a[0..m]$都加了 1，但是如果是②方式转移而来的话是不需要加那个 1 的，所以我们要用 $a[k]-1$ 来更新），也就是执行 $a[j]=\min(a[j],a[k]-1)$操作，这可以很容易地用线段树完成。$dp[i][j]$求解完成后，$a[0]+i-1$ 就是 $dpl[i]$的值了。

同理，我们求出 $dpr[i]$的值。

对于一个点 i，它不需要道路就可以满足条件的充要条件是 $dpl[i]=dpr[i]=0$。我们记录一个前缀和 $sum[i]$表示前 i 个点有多少个不需要添加道路就能满足条件。然后枚举 L 的位置，随着 L 的增加，R 的最优位置也一定是增加的，因此我们只需要用 $O(n)$的时间即可完成答案的更新，对于一对 L 和 R，我们用 $R-L+1-(sum[R]-sum[L-1])$来更新答案即可。时间复杂度 $O(m+(n+k)\log m)$，其中 k 为已知的横向道路条数。

1108：[POI2007]天然气管道 Gaz

直接贪心搞一下就行了，把黑点白点都按照 x 降序排列，对于一个黑点，我们去找可行的白点中 y 坐标小于等于黑点 y 坐标的最大的那个就可以了，用一个平衡树维护即可。

1109：[POI2007]堆积木 Klo

定义状态 $dp[i]$表示第 i 个位置满足要求，能满足的最大个数。则 $dp[i]=\max(dp[j])+1$，其中需要满足：

$0<=j<i$①，

$a[j]<a[i]$②，

$i-a[i]>=j-a[j]$③。

我们记 $g[i]=i-a[i]$，对于所有 $g[i]<0$ 的都是不合法的。这样做时间复杂度为 $O(n^2)$。考虑优化上面的方程，我们发现对每个 i 来讲，如果满足条件②的话，那么 $a[j]$一定小于 $a[i]$，因此我们可以考虑按照 $a[i]$递增的顺序进行动态规划。重新排序 n 个数，以 $a[i]$为关键字增序排列，并记录每个数原来的编号是多少。我们发现如果位置和数值确定，那么 $g[i]$也就确定。因此方程变成 $dp[i]=\max(dp[j])+1$，其中 $a[j]<a[i]$④，$g[j]<=g[i]$⑤。我们可以发现如果满足了④和⑤，那么一定可以满足①②③，这个通过联立④⑤就可以证明。由于我们已经按照 $a[i]$递增的顺序排列了，因此方程可以继续化简为 $dp[i]=\max(dp[j])+1$，$g[j]<=a[i]$。这个可以用树状数组或者线段树维护一下最大值就可以了。时间复杂度 $O(n\log n)$。

1110：[POI2007]砝码 Odw

我们考虑对 m 个砝码进行排序，则 m_{i+1} 一定是 m_i 的倍数。又因为最大的砝码重量不超过 10^9，因此最多只有 30 种不同的砝码。我们考虑将每个容器分解，按照每种砝码重量从大到小的顺序进行分解，这样最后统计出了第 i 种砝码可以放 $c[i]$个。由于要求答案最大，因此我们一定是从小到大加入每个砝码，对于当前的一个砝码

如果 $c[i]$不为 0，则把 $c[i]$减 1，答案加 1。否则就去借更大的 $c[j]$，我们记 j 的重量为 a，i 的重量为 b，则我们可以把 1 个 a 换成 a/b 个 b，如果可以找到这样的 j，则借过来一个答案加 1。

1111：[POI2007]四进制的天平 Wag

我们首先考虑把 n 转换成一个四进制数。我们记转换完的数有 m 位，第 i 位（从低到高）的值为 $num[i]$。我们发现对于第 i 位要想满足条件只有两种放置方法，第一种是在天平的右盘放 $num[i]$个 4^i的砝码，第二种是在天平的右盘放 1 个 4^{i+1}的砝码，在天平的左盘放 $4-num[i]$个 4^i的砝码。因此我们可以考虑用 DP 解决，定义状态 $dp[i][j]$表示第 i 到 m 位已经处理完，$dp[i][0]$表示第 i 位不多放一个在右盘，$dp[i][1]$表示在第 i 位多放一个在右盘，最少需要放置的砝码个数。则有：

$dp[i][0]=\min(dp[i+1][0]+num[i],dp[i+1][1]+4-num[i])$;

$dp[i][1]=\min(dp[i+1][1]+num[i]+1,dp[i+1][1]+3-num[i])$;

对于方案数，只要在动态规划的过程中顺便处理一下就可以了。

1112：[POI2008]砖块 Klo

我们首先考虑让 $a[1..k]$变成一样的，我们不妨设最后相同的值为 p，则代价为 $\sum_{i=1}^{k}|a[i]-p|$。易知在 p 为 $a[1..k]$的中位数时，代价取到最小值。这样我们就得到了当左端点在 1 时的答案，接下来如考虑如果知道了左端点在 i 的答案，如何求出左端点在 $i+1$ 的答案。我们不难发现其实 $a[i..i+k-1]$和 $a[i+1..i+k]$有很大的一段交集，不同的只有两个数。因此我们需要一个数据结构能够支持：

①．插入一个数

②．删除一个数

③．询问中位数

④．询问比中位数小的所有数的和，询问比中位数大的所有数的和。

我们可以考虑使用平衡树进行维护，只要把小于等于中位数的建成一棵平衡树，把大于中位数的建成一棵平衡树，然后维护即可。时间复杂度 $O(n\log k)$。

1113：[Poi2008]海报 PLA

宽度没有用，高度用单调栈维护。时间复杂度 $O(n)$。

1116：[POI2008]CLO

原图可能不连通，我们考虑处理每个连通分量，如果一个连通分量无解，那么就是无解。对于一个连通分量，如果这个连通分量是一棵树则显然无解，因为根节点必然没有入度来源。否则先随便求一个生成树记为 T，找到不在 T 中的一条边，记这两个点为 A 和 B，那么我们只需要以 A 为根节点，重建这棵树，这样每个点的入度来源就是它的父亲节点，根节点的入度来源就是 B。时间复杂度 $O(n+m)$。

1121：[POI2008]激光发射器 SZK

我们考虑给定的每个点发出的光线的方向，方向只有四种情况，至于具体是哪一种，这取决于与连接这个点的两条边的方向，这个打一个表即可，不同的方向组合一共有 8 种。分别是(向上转向左，右上)，(向上转向右，

右下)，(向下转向左，左上)，(向下转向右，左下)，(向左转向上，右上)，(向左转向下，左上)，(向右转向上，右下)，(向右转向下，左下)。然后我们处理每个点（不光是输入的点，还有多边形上的其它点）沿着四个方向走，下一个最近的是哪个点。这个只要按照 $x+y$，$y-x$ 排序然后乱搞一下就可以了。至于答案，第一问的答案显然就是 $n/2$。对于第二问，我们考虑到由于光路可逆，设由点 i 反射到点 j，经过了 $d_{i,j}$ 次反射。易知所有的 $d_{i,j}$ 之和应该是和多边形周长同阶的，因此我们只要模拟每个点的走法就可以了。时间复杂度 $O(n+L\log L)$。

1122：[POI2008]账本 BBB

我们考虑枚举操作②的次数，这样我们可以把原序列 S 复制成 SS。如果移动 0 次，那么则对应 $S[n+1,2n]$。我们考虑记录最小的前缀和，那么我们根据这个前缀和就可以算出此次的代价，然后由 $n+1$ 转移到 n，然后不断向前转移 n 次，只需用单调队列维护前缀和即可。时间复杂度 $O(n)$。

1123：[POI2008]BLO

考虑以 1 为根节点开始进行 *dfs*，我们记 $ans[i]$表示删掉点 i 以后，有多少对$(i,j)(i<j$ 且点 i 和点 j 不连通)。利用 *tarjan* 求割点的过程，如果一个点不是割点，则 $ans[i]=0$，否则的话记以 i 为根 *dfs* 得到的子树大小依次为 $a[1]$，$a[2]$，$a[m]$，$s[i]=\sum_{j=1}^{i}a[j]$，则 $ans[i]=\sum_{j=2}^{m}s[j-1]\times a[j]+s[m]\times(n-s[m]-1)$，根节点的值特判一下，如果根节点不是割点的话，则把 $ans[1]$赋值成 0。我们输出的值则为 $2\times(ans[i]+n-1)$。时间复杂度 $O(n+m)$。

1124：[POI2008]枪战 Maf

原图可能存在多个连通分量，由于不同连通分量之间的答案互不影响，我们考虑单独的一个连通分量。我们可以发现每个连通分量都是一个内向树，记 $next[i]$表示 i 瞄准的人。对于一个连通分量，最少死多少人我们可以找一个入度为 0 的点 i，然后 i 开枪射死 $next[i]$，不断找下去，直到找不到入度为 0 的点。此时就有两种情况，如果环上的点一旦被射死，那么环上的点就和普通的点一样了。否则我们记环上有 m 个点，那么显然最少再死亡$\left\lfloor\frac{m+1}{2}\right\rfloor$个点。对于最多死亡多少人，如果这个连通分量仅仅只是一个环的话，如果环上有 1 个点，那么就是最多死 1 人，否则如果环上有 m 个点，那么就是最多死 $m-1$ 个人。如果这个连通分量不仅仅是一个环的话，我们可以对每棵树贪心处理，我们先不管每个根节点（环上的点）的儿子，让别的儿子节点依次射死对应父亲节点，这样最后就剩下的就是所有入度为 0 的点，然后我们考虑一个根节点 p，p 射死 $next[p]$，然后 p 的儿子射死 p，这样环上的点只要按照破开成的链的顺序依次射击就可以全部死亡，也就是这种情况下，我们记一共有 a 个点，环上有 m 个点，入度为 0 的点有 b 个，则最多可以死亡$(a-m)-b+m=a-b$ 个点。时间复杂度 $O(n)$。

1125：[POI2008]Poc

我们考虑把每个字符串做个 *hash*，记第 i 个字符串的第 j 位为 $str[i][j]$，则 $hash[i]=\sum_{j=1}^{len}seed^j\times str[i][j]$。我们把所有操作提取出来，包括最初的给定 n 个字符串，视为 n 个插入操作。对于 m 个修改，每个修改视为 2 个插入操作和 2 个删除操作，注意如果是交换字符的两个串是同一个串，则视为 1 个插入和 1 个删除。最后把每个字符串的最终状态视为 n 个删除操作。按照 *hash* 为第一关键字，时间为第二关键字的顺序进行排序。我们将所

有的操作进行分组，把所有 *hash* 相同的放在一组。对于一个字符串 *i*，在插入的时候把它插入，删除的时候更新答案。记 *sum*[*i*]表示时刻 *i* 的字符串个数(0<=*i*<=*m*+1)，第 *i* 个字符串的插入时间为 *st*[*i*]，删除时间为 *en*[*i*]。则我们用 max(*sum*[*st*[*i*]],*sum*[*st*[*i*]+1]..*sum*[*en*[*i*]-1])来更新第 *i* 个串的答案，用个线段树就行了。

1127: [POI2008]KUP

首先如果存在一个元素在值在[*K*,2*K*]之间，那么我们只需要把它输出即可。下面我们考虑每个格子的数都小于 *K* 或者大于 2*K* 的情况。我们把所有大于 2*K* 的格子作为障碍，然后找到一个最大的子矩阵，下面我们证明如果题目有解则这个最大的子矩阵内部一定存在解。我们一行一行扫描最大的子矩阵：

①如果当前行的和在[*K*,2*K*]，则输出当前行作为答案；

②如果当前扫过的所有行总和在[*K*,2*K*]则输出当前这些行作为答案；

③如果当前行的权值大于 2*K*，则当前行一定可以构造出解。我们考虑一行数，每个数要么是小于 *K* 要么大于 2*K*，一个一个扫描这行的数，假定当前扫描到的和为 *S*，如果 *S*<*K* 则继续扫描下一个，直到第一次 *S* 大于等于 *K* 为止，假设 *S* 可以从小于 *K* 直接变成大于 2*K*，则这与每个数都小于 *K* 矛盾，因此如果一行的数大于 2*K*，则这行一定可以构造出解。时间复杂度 $O(n^2)$，内存卡的比较严。

1128: [POI2008]Lam

由于前面的操作可能会被后面的操作覆盖，因此我们不妨考虑倒着计算每个答案。记输入的第 *i* 个数为 *p*[*i*]。则有：

$$ans[n]=\frac{1}{p[n]},$$

$$ans[i]=\frac{1}{p[i]}\times\prod_{j=i+1}^{n}\frac{p[j]-1}{p[j]},$$

$$ans[i+1]=\frac{1}{p[i+1]}\times\prod_{j=i+2}^{n}\frac{p[j]-1}{p[j]},$$

则 $\frac{ans[i+1]}{ans[i]}=\frac{p[i]}{p[i+1]-1}$，也就是 $ans[i]=\frac{p[i+1]-1}{p[i]}\times ans[i+1]$。

然后这个用高精度算一下就可以了，涉及到高精度乘单精度，高精度除单精度，高精度取模单精度。注意一下如果 *p*[*i*]=1，则对于任意的 *j*<*i* 有 *ans*[*j*]=0。

1130: [POI2008]POD Subdivision of Kingdom

爆搜就可以了，我们考虑把每个点分到集合 *A* 还是集合 *B*，用二进制表示集合内的点，既然要求最少删多少条边，我们可以等价去求最多保留多少条边。对于一个点 *p* 加入集合 *A*，可以新增的边数是与 *p* 相邻的点组成的集合与集合 *A* 的交集，这就涉及求一个数的二进制有多少个 1，如果暴力预处理的话可能会爆空间。我们可以考虑处理 2^{13} 个，记 *c*[*i*]表示 *i* 的二进制有多少位，则一个数 *x* 对应的结果就是 *c*[*x*>>13]+*c*[*x*&8191]。时间复杂度 $O(2^n)$。

1131: [POI2008]Sta

题目要为一棵无根树树找一个根节点，使得所有点的到我们找的这个根节点的距离之和最大。我们先以 1

为根节点进行一次树形 DP。记 *size*[*i*]表示以 *i* 为根的子树的节点个数（包括 *i*），*down*[*i*]表示 *i* 的子树的所有节点到 *i* 的距离和，*up*[*i*]表示非 *i* 的子树的节点到 *i* 的距离和。我们通过一次 DP 自底向上先求出 *size*[*i*]和 *down*[*i*]，然后自顶向下求出 *up*[*i*]，转移方程是：

$size[p]=\sum_{v\in son[p]} size[v]+1$，

$down[p]=\sum_{v\in son[p]} down[v]+size[p]-1$，

$father[p]=v$，

$$up[p]=up[v]+n-size[v]+down[v]-\big(down[p]+size[p]\big)+size[v]-size[p]$$
$$=up[v]+n+down[v]-down[p]-2\times size[p]$$

。

只有 *up* 的推倒比较麻烦一下，自己画画就知道了。时间复杂度 *O*(*n*)。

1132: [POI2008]Tro

我们考虑枚举三角形的左下角顶点，具体做法是把 *n* 个点进行排序，按照 *x* 为第一关键字升序，*y* 为第二关键字升序。对于第 *i* 个点，我们固定三角形的一个顶点是点 *i*，剩下的两个点在 *i*＋1 到 *n* 中选择，记 *m*＝*n*－*i*。则我们把这 *m* 个点取出，把第 *i* 个点作为原点，平移这 *m* 个点，然后把这 *m* 个点进行极角排序。根据向量叉积，以点 *i* 为左下角的三角形面积和为 $\frac{\sum_{i=1}^{m}\sum_{j=i+1}^{m}\left(x_i y_j-x_j y_i\right)}{2}$，我们考虑分子的求和，记 $sumx_i=\sum_{j=1}^{i}x_j$，$sumy_i=\sum_{j=1}^{i}y_j$，则 $\sum_{i=1}^{m}\sum_{j=i+1}^{m}\left(x_i y_j-x_j y_i\right)=\sum_{i=1}^{m}\left[x_i\times\left(sumy_m-sumy_i\right)-y_i\times\left(sumx_m-sumx_i\right)\right]$。时间复杂度 *O*($n^2\log n$)。

1145: [CTSC2008]图腾 totem

我们定义 *f*(*abcd*)表示高度排名为 *abcd* 的个数，如 *f*(1324)就表示题目所求的闪电。则我们所求为 *f*(1324)－*f*(1243)－*f*(1432)。其中 *f*(1324)＝*f*(1X2X)－*f*(1423)，*f*(1243)＝*f*(12XX)－*f*(1234)，*f*(1432)＝*f*(14XX)－*f*(1423)。则有 *f*(1324)－*f*(1243)－*f*(1432)＝*f*(1X2X)－*f*(1XXX)＋*f*(13XX)＋*f*(1234)。我们对这四项逐一考虑：①*f*(1XXX)，这个最好求了。比如当前位于第 *i* 个，*i*＋1 到 *n* 中比 *a*[*i*]大的数有 x 个。则 *f*(1XXX)＝*C*(x,3)。②*f*(1234)，对于这个我们维护 3 个树状数组，其中 $C1_i=\sum_{j=i+1}^{n}1\left(a_j>a_i\right)$，$C2_i=\sum_{j=i+1}^{n}C1_j\left(a_j>a_i\right)$，$C3_i=\sum_{j=i+1}^{n}C2_j\left(a_j>a_i\right)$。这三个量都可以在倒序扫描中快速求出。③*f*(13XX)，这个我们可以在 3 的位置统计。首先右侧一定有一个是 4，那么这个个数为 $C1_i$，然后我们考虑 132 的情况。我们可以发现 132＝(132＋312＋321)－(312＋321)。对于第二个括号显然为 $C\left(C1_i,2\right)$。我们考虑第一个括号，我们设当前点的纵坐标为 *x*，那么所有的 *x* 坐标一定为 1~*n* 的一个排列，那么比 *x* 小的一定有 *x*－1 个。因此我们用树状数组维护 *x*－1 的和即可。④*f*(1X2X)，这个我们在 2 的位置进行统计，那么显然第 2 个 X 可以填的数的个数为 $C1_i$，而 1X2，显然 X 要比 1 和 2 都大，我们可以考虑两种情况，第一种是 X 满足比 1 和 2 都大，第二种是不满足。那么我们可以通过求出第一种和第二种的和然后减去第二种的来求出 1X2 的答案，具体实现和③类似。

1146: [CTSC2008]网络管理 Network

这个题好麻烦，真心膜拜能在现场写出 AC 代码的人。一棵树，两个操作，1 修改某一个点的点权，2 问两点间的第 K 大点权。做法就是树链剖分然后树套树，这题用动态树貌似实现不了。

1149: [CTSC2007]风玲 Mobiles

记 $low[i]$表示以 i 为根的子树深度的最小值，$high[i]$表示以 i 为根的子树深度的最大值，对于一个点 p，我们递归处理 $lch[p]$和 $rch[p]$，然后合并 $lch[p]$和 $rch[p]$的信息并进行判断即可。注意对于一个合法解树是深度一定不会特别大，加上这个特判以防递归爆栈。

1150: [CTSC2007]数据备份 Backup

每条边一定的编号为奇数的点连向编号为偶数的点，因此可以考虑建立二分图，然后求最小费用最大流，但这个做法显然超时。一个观察就是增广路只有两种，距离考虑一下，然后用堆或者平衡树维护即可，时间复杂度 $O(n\log n)$。

1179: [Apio2009]Atm

首先进行强连通缩点，维护每一个连通分量中，是否有一个点是酒吧，以及总钱数的多少。对缩点后的图进行拓扑排序，然后按照拓扑排序进行动态规划。记 $dp[i]$表示走到点 i 的最大获利，对于给定的起点属于的连通块有 $dp[i]=money[i]$，其余点的初值都是 $dp[i]=$负无穷。对于一条边 i 连接 j，我们用 $dp[i]+money[j]$来更新 $dp[j]$。答案是满足一个连通块中至少有一个是酒吧的强连通分量中最大的 $dp[i]$。由于会爆栈，强连通分量算法要进行人工堆栈。时间复杂度 $O(n+m)$。

1180: [CROATIAN2009]OTOCI

直接用 LCT 做就可以了。

1191: [HNOI2006]超级英雄 Hero

我们设答案为 Ans，则前 Ans 个都必须要取到，所以我们可以二分答案，然后用二分图最大匹配或者网络流检验。

1193: [HNOI2006]马步距离

首先把起点平移到(0,0)，然后对应地平移终点，设平移以后终点是(x,y)，我们可以发现 x 和 y 的正负对结果并无影响，因此我们假设整个算法的坐标始终都是非负的。由于 x 和 y 可能很大，因此我们可以贪心地走一段时间，如果 x 比 y 大，则 x 减小 2，y 减小 1，答案加 1，否则 x 减小 1，y 减小 2，答案增加 1。重复做上面的过程直到 x 和 y 都在 20 以内，然后再从(0,0)开始到现在的(x,y)进行 BFS 就可以了。

1195：[HNOI2006]最短母串

直接状压 DP 解决，开一个 *pair<int,string>*类型的数组，记 *dp*[*i*][*j*]表示已经满足的字符串用二进制表示为 *i*，其中最后一个字符串是 *j* 时的最短长度，在最短长度前提下的最小字典序串是什么。预处理第 *i* 个串后面接第 *j* 个串，能重合的最长字母为多长，也就是最少需要的 *j* 的后缀长度多长。然后用 *dp*[*i*][*j*]去更新 *dp*[*i*|1<<*k*][*k*]即可。

1196：[HNOI2006]公路修建问题

我们二分最终答案，对于当前的 *mid*，用并查集类似于 Kruskal 的方法进行判断，检验方法如下：首先尽可能连接一级道路，如果这个都无法满足要求的话那就说明是不符合的，接下来连接二级道路，如果最后可以连出 *n*－1 条则是符合的，否则是不符合的。

1197：[HNOI2006]花仙子的魔法

找规律，*dp*[*i*][*j*]表示 *i* 维空间使用 *j* 次魔法，则 *dp*[*i*][0]＝1，*dp*[1][*i*]＝2×*i*，*dp*[*i*][*j*]＝*dp*[*i*][*j*－1]＋*dp*[*i*－1][*j*－1]。

1202：[HNOI2005]狡猾的商人

对于一个信息 *a,b,c* 我们连边 *a*－1 到 *b* 权值为 *c*，*b* 到 *a*－1 权值为－*c*，设置超级源点 *S* 向每一个点连权值为 0 的边，显然存在负环则无解。

1211：[HNOI2004]树的计数

有解的情况，设每个点的度为 *d*[*i*]，根据 *prufer* 编码，答案是 $\frac{(n-2)!}{\prod_{i=1}^{n}(d[i]-1)!}$。无解的有两种情况，①一个点的树，这个度不为 0。②不为一个点的树，存在一个点的度小于等于 0，或者 $\sum_{i=1}^{n}(d[i]-1)\neq n-2$。

1218：[HNOI2003]激光炸弹

这道题我们用扫描线及线段树进行维护。我们可以发现对于边界上的点不算这个条件比较麻烦。我想到的解决办法就是变成算左边界和下边界，而不算上边界和右边界。不难发现这样做原问题的答案没有影响。我们首先把所有坐标加 1，使得所有坐标都是正数，记录 *maxx* 表示最大的横坐标，*maxy* 表示最大的纵坐标。具体实现是把所有坐标按照 *x* 为关键字排序，用一个从垂直于 *x* 轴的扫面线从 *x*＝1 开始扫描到 *maxx*，并且维护一个队列，保存当前满足横坐标在 *x* 到 *x*＋*R*－1 范围内的所有点（*R* 是那个正方形的边长）。然后我们抽象出一个序列，其中 *a*[*i*]表示纵坐标在[*i*,*i*＋*R*－1]范围内所有点的价值和，容易发现插入一个点实际是相当于线段树的区间修改，在线段树的每一个节点记录一个最大值并且打一个叫 *add* 的 *lazy* 标记就可以了。当前扫面线对应正方形的左边界的最大值显然就是根节点的最大值数据。每个点最多插入一次删除一次。时间复杂度为 *O*(*n*log*n*＋*n*log*maxy*)。

1221: [HNOI2001] 软件开发

将每一天拆成两个点 i 和 i'，设置附加源点 S，附加汇点 T，设第 i 天需要 p 个毛巾。S 向 i 连容量为 p 费用为 0 的边（满足个数够），i 向 T 连容量为 p，费用为 f 的边（直接买新毛巾），i'向 T 连容量为 p 费用为 0 的边（通过别的方式得到毛巾），如果 $i\neq n$，则 i 向 $i+1$ 连容量为＋∞费用为 0 的边（上一天没用完的下一天接着用），如果 $i+a+1<=n$，则 i 向$(i+a+1)'$连容量为＋∞费用为 fa 的边（A 种方式获得毛巾），如果 $i+b+1<=n$，则 i 向$(i+b+1)'$连容量为＋∞费用为 fb 的边（B 种方式获得毛巾），求出最小费用最大流即可。

1225: [HNOI2001] 求正整数

显然答案包含的质因子最多只能是前 15 个素数，爆搜每个素数的次幂，由于高精度比较大小很慢，所以可以换成以某个数为底的对数进行比较，然后记录每个素数的次幂，写一个高精度乘起来就可以了。

1227: [SDOI2009]虔诚的墓主人

把 y 坐标离散化，按照 x 相同的分段，预处理一个组合数，然后用树状数组统计。有一个小技巧，由于取模的数是 2 的次幂，所以我们可以不取模，让他任意溢出，因为这并不影响结果，只需要最后把负的答案加成正的即可。

1230: [Usaco2008 Nov]lites 开关灯

简单的线段树问题，在节点打一个 rev 标记表示当前节点是否取反，然后注意维护每个节点的 sum 就可以了。

1231: [Usaco2008 Nov]mixup2 混乱的奶牛

由于 $n\leqslant 16$。很容易想到状压 DP，用一个 16 位的二进制储存每个牛是否用过。时间复杂度 $O(n^2\times 2^n)$。

1232: [Usaco2008Nov]安慰奶牛 cheer

我们考虑这样一个问题。对于一条边，如果想走的话，那么代价就是这条边连接的两个点的权值加上二倍的这个边的权值。这样我们就把所有的点权都加载到了边上。求一遍最小生成树。下面决策起点，其实起点只要选择点权最小的点就可以了，把最小生成树的答案加上这个点权就是最后的答案了。不懂怎样证明，样例看起来就是这样的，而且能通过所有测试数据。

1233: [Usaco2009Open]干草堆 tower

这道题需要发现必存在一种最优的放置方法，使得底层的宽度最小，这个证明我也不会。然后就可以 DP 了，DP 的过程中记录一下第 i 的干草所处的高度，$dp[i]$表示$[i,n]$区间内的干草安排完成，第 i 个干草所在的层的最小宽度，我们记 $sum[i]=w[1]+w[2]+\ldots+w[i]$。则有 $dp[i]=\min(sum[j-1]-sum[i])$其中 $sum[j-1]-sum[i]>=dp[j]$。对这个方程用单调队列优化就可以了。

1251：序列终结者

裸题，SPLAY 维护。

1257：[CQOI2007]余数之和 sum

显然 $ans=\sum_{i=1}^{n}k \bmod i=nk-\sum_{i=1}^{n}(k \operatorname{div} i)\times i$。$k \operatorname{div} i$ 的取值个数在 $O(\sqrt{n})$ 数量级。所以可以在 $O(\sqrt{n})$ 复杂度下解决。

1263：[SCOI2006]整数划分

尽量分成 3，如果剩下了 1，那么拿出一个 3，把 1 和 3 变成两个 2，必然使结果最优。也就是对于输入的 n，我们设 $ans=3^a\times 2^b$，如果 n 除以 3 余 1，$a=n/3-1,b=2$，如果 n 除以 3 余 2，$a=n/3,b=1$，如果 n 整除 3，$a=n/3,b=0$。

1269：[AHOI2006]文本编辑器 editor

这题很常规啊，就是个 Splay。开两个哨兵，一个放在字符串的最左端，一个放在最右端，注意及时下传标记就可以了。

1270：[BeijingWc2008]雷涛的小猫

由于不管怎么跳，高度都在减小，所以用 DP 解决。状态 $dp[i][j]$表示所处高度 i，第 j 棵树上能吃到的最多数量。

第一种情况是直接从当前树的上面跳下来，第二种情况是从别的树上斜着跳下来，我们记 $dpmax[i]=\max(dp[i][j])$。则有 $dp[i][j]=\max(dp[i+1][j],dpmax[i+d])(i+d<=m)+a[i][j]$，$a[i][j]$表示当前位置的数量。答案就是 $dpmax[0]$，时间复杂度 $O(nm)$，n 是树的个数，m 是最大高度。

1293：[SCOI2009]生日礼物

首先按出现位置排序。记 $L=1$，当前维护的点为 i，如果 i 的颜色没出现过则把颜色总数加 1，如果 L 位置的颜色出现次数大于 1 则把 L 增大到颜色数等于 1，如果 L 到 i 的颜色总数等于 k，则有 i 的位置减去 L 的位置来更新答案。

1295：[SCOI2009]最长距离

分别以每个点为源点 S 求单源最短路，定义格子 u 到格子 v 的边权，如果 v 有障碍，边权是 1，否则边权是 0。如果一个点和 S 的距离小于等于 T（题中所说的障碍总数），则用这个点到 S 的距离更新答案。注意一下如果源点是障碍格子的话，那么初始距离要赋成 1 而不是 0。

1296：[SCOI2009]粉刷匠

首先可以发现空着不染是没有意义的，因为等价于染错了。所以我们可以认为求染满整个格子最多能染多少个正确的。$dp[T][i][j]$表示染了 T 次，到达第 i 行第 j 列，最多正确染多少个。则有：$dp[T][i][0]=dp[T][i-1][m]$，$dp[T][i][j]=\max(dp[T-1][i][k]+getsum(i,k+1,j))$。其中 $getsum(i,j,k)$表示第 i 行染第 j 个方块到第 k 个方块最多能正确多少个，这个值等于这一部分方块中为 0 的和为 1 的个数较大的那个。

1297：[SCOI2009]迷路

容易看出是矩阵乘法，但是直接做比较麻烦。由于每条边的长度不超过 9，我们可以考虑把每个点进行拆点。对于第 i 个点，我们把它拆成 9 个，依次是 $9\times(i-1)+j$，$1<=j<=9$。对于一条边从 i 到 j 边权为 k，我们把拆点后的图的第 $9\times(i-1)+k$ 到 $9\times(j-1)+1$ 连一条边，这个意思是在第 i 个点停留 $k-1$ 次，然后前进一次到底第 j 个点。最后我们把 $9\times(i-1)+j$ 到 $9\times(i-1)+j+1$，$1<=j<=8$ 连一条边，表示在自己停留是合法的。然后求出这个矩阵的 T 次方，再和一个一行的矩阵相乘即可，一行矩阵的初值是除了第一个元素的 1，其余元素都是 0。最终答案就是这个矩阵的第 $9\times(n-1)+1$ 格子上面的数。

1303：[CQOI2009]中位数图

我们设中位数为 b，首先我们找到 b 的位置我们设为 p。接下来找出对于任意位置 i，它到位置 p 中的所有数中，比 b 大的减去比 b 小的有多少个，我们分别把 p 的左边和 p 的右边（左边右边都包 p）进行统计。假设 p 的左边大减小差为 k 的有 a 个，p 的右边大减小差为 $-k$ 的有 b 个，那么我们把答案加上 $a\times b$ 即可。时间复杂度 $O(n)$。

1305：[CQOI2009]dance 跳舞

网上有神贪心做法，但是我没有想到。用网络流二分答案解决。对于当前的 mid，检验方法如下：我们设每个男生最多和自己不喜欢的女生跳 lim 次，把每个男生 i 拆点 a，b，c。其中 a 表示大局，b 表示和自己喜欢的跳，c 表示和自己不喜欢的跳，女生同理。设源点为 S，汇点为 T。S 向每个男生的 a 连容量为 mid 的边，每个 a 向 b 连容量正无穷的边，向 c 连容量 lim 的边，女生同理。对于喜欢的关系由男生的 b 向女生的 b 连容量为 1 的边，对于不喜欢的关系由男生的 c 向女生的 c 连容量为 1 的边。如果最大流等于 $mid\times n$ 则可行，否则不可行。

1307：玩具

乱搞吧，我觉得我的算法也并不正确，连 $n=5$，1 2 3 5 4 这种都算不对，但能 AC。

1334：[Baltic2008]Elect

按数从大到小排序，然后背包。记总和为 s，总和的一半为 p，$dp[j]$表示在和不超过 j 的情况下，能得到的最大值。考虑第 i 个数 $a[i]$，当 $j>=a[i]$时，若 $dp[j-a[i]]<=p$ 且 $dp[j-a[i]]+a[i]>p$ 则用 $dp[j-a[i]]+a[i]$更新 $dp[j]$。若 $dp[j-a[j]]+a[i]<=p$ 则用 $dp[j-a[i]]+a[i]$更新 $dp[j]$。时间复杂度 $O(ns)$。

1336: [Balkan2002]Alien 最小圆覆盖

用随机增量最小圆覆盖，处理好怎么解一个三角形的外心就行了。

1337: 最小圆覆盖

同 1336。

1406: [AHOI2007]密码箱

若 $x^2 \equiv 1(\bmod n)$，则有 $n \mid (x-1)(x+1)$，我们不妨设 $n=ab$ 且 $a \mid (x-1)$，$b \mid (x+1)$。我们考虑枚举 a 的值，则 x 一定可以写成 $a+1+ka$，我们找 $a+1+ka$ 能整除 b 的作为答案即可，然后再枚举一下 b，判重输出就可以了。时间复杂度 $O(\sqrt{n}\log n)$。

1407: [Noi2002]Savage

由于答案不会很大而 n 又非常小，我们考虑枚举答案判断是否可行。记当前枚举的答案为 T，对于第 i 个人和第 j 个人，如果存在一个整数 x 满足 $0 \le x \le \min(L[i], L[j])$ 且 $C[i]+xP[i] \equiv C[j]+yP[j](\bmod T)$，则当前的答案 T 不可行。因此我们只需要求出 $C[i]+xP[i] \equiv C[j]+yP[j](\bmod T)$ 的最小整数解或者判断它没有解即可。化简一下有 $(P[i]-P[j])x \equiv C[j]-C[i](\bmod T)$。这个用扩展欧几里得解一下就可以了。

1415: [Noi2005]聪聪和可可

记忆化搜索。记状态 $dp[i][j][0]$表示聪聪在点 i，可可在点 j，且这一步是聪聪走，聪聪抓到可可的期望步数，类似定义 $dp[i][j][1]$。则有 $dp[i][j][0]=dp[next[i][j]][j][1]+1$，$dp[i][j][1]=\Sigma(dp[i][j'][0])+dp[i][j][0]/(d+1)$，其中 j' 表示与点 j 相邻的点，d 表示与点 j 相邻的点有多少个。

1432: [ZJOI2009]Function

显然当函数都为一次函数时答案取得最小值。如果 $n=1$ 那么 $ans=1$，否则 $ans=\min(k, n-k+1)\times 2$。

1433: [ZJOI2009]假期的宿舍

简单的网络流问题，把每个学生拆成俩点，表示是否回家，是否在宿舍。S 向不回家的连容量为 1 的边，住宿舍的向 T 连容量为 1 的边。左边向右边连边，前提的得认识。如果最大流等于不回家的人数则可行，否则不可行。

1443: [JSOI2009]游戏 Game

首先忽略不能到达的格子，然后对整个棋盘进行黑白染色。求一次二分图最大匹配，记这个值为 T。考虑一个点，如果删掉以后最大匹配不变，则这个点是必胜点，否则不是。对于判断过程只需把匹配它的点进行标记，然后重新增广一下即可，证明我也不太清楚。

1449: [JSOI2009]球队收益

我们发现对于任意一个选手 i，题目最初给出的信息中赢了 $win[i]$场，输了 $lose[i]$场，他共参加了 k 比赛，那么所有比赛结束以后他的新 $win[i]$加上新 $lose[i]$应该是等于 $win[i]+lose[i]+k$ 的，因此我们可以考虑将 $lose[i]$用 $win[i]$来表示。

设一共赢了 x 场（这个不算最初赢的那些场），则有 $W=c(win+x)^2+d(lose+k-x)^2$，我们不妨考虑将 $lose$ 用 $lose+k$ 来代替，则有 $W=c(win+x)^2+d(lose-x)^2$。展开这个表达式有 $W=(cwin^2+dlose^2)+(2cwin-2dlose)x+(c+d)x^2$。我们可以发现第一项为定值，因此我们可以先不考虑它，最后把每个人都加上那个值即可。我们考虑如何最小化 $\sum((2cwin-2dlose)x+(c+d)x^2)$，且满足 $\sum x=m$。我们考虑对于一个球员，当他的赢的场数 x 由 $i-1$ 变成 i 时收益的增量为多少，显然这个值为 $2cwin-2dlose+(c+d)\times(2i-1)$。由于 c 和 d 都是非负的，因此随着 i 的增加，这个增量也在单调增加。所以一个球员参加 x 场比赛所对应的增量刚好就是 i 取 1 一直到 x 增量的总和。我们考虑建立费用流模型。

添加附加源点 S，附加汇点 T。二分图的左侧放球员，右侧放比赛。每场比赛向 T 连容量为 1，费用为 0 的边。每个球员向他参加的所有比赛连容量为 1 费用为 0 的边。对于一个球员 i，记他一共参加了 x 场比赛（不算最初赢和输的那些场）。则对于任意的 $1\le j\le x$，由 S 向 i 连容量为 1 费用为 $2cwin-2dlose+(c+d)\times(2j-1)$ 的边。求出最小费用最大流即可。

1452: [JSOI2009]Count

由于颜色数很少，对于每一个颜色建一个二维树状数组就可以了。

1458: 士兵占领

我们考虑这个问题的反面。首先我们把整个棋盘放满，如果这都不行的话那就直接输出 JIONG!了。否则我们继续考虑如何去掉最多的棋子使得满足要求，我们根据满的棋盘算出每行多放了多少并记为 $c1[i]$，每列多放了多少并记为 $c2[i]$。接下来用网络流解决，建立附加源点 S，附加汇点 T，S 向每行对应的点连容量为 $c1[i]$的边，每列对应的点向 T 连容量为 $c2[i]$的边，如果一个格子(i,j)是无障碍的，则从第 i 行对应的点向第 j 列对应的点连容量为 1 的边。然后用总数减去最大流就可以了。

1485: [HNOI2009]有趣的数列

打表发现，答案就是 *Catalan* 数列的第 *n* 项。

1486：[HNOI2009]最小圈

简单 01 分数规划问题，我们考虑二分答案，对于当前二分等待检验的值 *mid*，我们这样检验。假设存在一个圈，我们设这个圈上的点依次为 c_1，c_2，c_3，c_4，…，c_k，其中 $c_{k+1}=c_1$。我们设相邻两点的距离为 $d(u,v)$，则有：$\frac{\sum_{i=1}^{k} d(c_i,c_{i+1})}{k} \le mid$，也就是 $\sum_{i=1}^{k} d(c_i,c_{i+1}) \le k \times mid$，整理得 $\sum_{i=1}^{k} \left[d(c_i,c_{i+1}) - mid \right] \le 0$，所以我们只要把所有的边权都减去 *mid*，这样就转变成了判原图有没有负环，用 SPFA 解决，正确的做法是如果一个点入队超过 *n* 次那就存在负环，但我的判法是如果入队总次数超过 10W 次就是有负环，这样正确的几率还是很高的，而且不会超时。

1491：[NOI2007]社交网络

用 Floyd 求最短路的同时更新点 *i* 到 *j* 的最短路个数，用乘法原理，然后用公式算一遍，时间复杂度 $O(n^3)$。

1494：[NOI2007]生成树计数

考虑到 *k* 很小，我们用状压 DP 解决。我们定义一种表示 *k* 个点的生成树方法，如 12234 表示共有 4 个连通块，第 2 和第 3 的点同属一个连通块，剩下的每个点属于一个连通块，我们对每一个连通状态都用这样的方法表示，并且用最小表示法，所谓最小表示法就是在所有表示同一联通状态的数中，最小的那个，比如我们用 12234 表示上面那个状态，而不用 13324。经过搜索发现当 $k=5$ 时，最多有 52 种不同的状态。我们在第一轮搜索的时候记录每个状态出现的次数。这样我们就求出了 *k* 个点时，所有的连通状态的个数。接下来我们考虑添加一个点，也就是变成 $k+1$ 个点。考虑两个状态 *A* 和 *B*。我们检验 *A* 状态能通过多少种方式变成 *B* 状态，我们不妨设 *A* 是 1 到 *k*，*B* 是 2 到 $k+1$，那么显然 *A* 中的点要是加边的话只能由 *A* 中的点连向 $k+1$，我们枚举每个点是否加边，然后用并查集判断连通性，需要满足的条件是点 1 必须能和 2 到 $k+1$ 的一个点相连，否则在 $k+2$ 的时候点 1 将永远无法和别的点相连，从而变成一个不合法的状态。其次 1 到 $k+1$ 组成的这个图应该是一个森林或者是一棵树，而不能是带环的东西。我们记录一个转移矩阵 $S[i][j]$，如果 *A* 可以转移给 *B*，我们则把 $S[A][B]$加 1，当然 *A* 转移给 *B* 的方式可能为 0，也可能大于 1。接下来我们考虑求出的初始的 *k* 个点的连通情况，我们设这个矩阵为 $D[1][i]$（显然这个矩阵只有 1 行），我们不难发现如果我们进行 $D \times S$ 的操作，就得到了 $k+1$ 的点的情况，如果进行 $D \times S \times S$ 就得到了 $k+2$ 的情况，也就是说我们只要把 *D* 乘上 $n-k$ 个 *S*，就可以得到 *n* 个点时的情况，这里用矩阵快速幂求就可以了。最后答案就是 11111 那个状态对应的值，因为最后要都连通才行。

1499：[NOI2005]瑰丽华尔兹

我们设 *n* 为行数，*m* 为列数，*p* 为有多少个区间，*t* 为总时间是多少。要想不超时的话那么时间复杂度就不能与 *t* 有关，也就是说我们只能按照区间逐一处理。状态 $dp[i][j][k]$表示在第 *k* 个区间的时候，走到第 *i* 行，第 *j* 列，能经过的最大格子数。我们假设第 *k* 个区间的行走方向为向上（其余三种情况可以同理）。则有转移方程 $dp[i][j][k]=\max(dp[i'][j][k-1]+i'-i)$。整理一下可得 $dp[i][j][k]=\max(dp[i'][j][k-1]+i') -i$。其中 $i'>=i$ 且 $i'-i$ 小于等于第 *k* 个区间的长度。我们发现 *j* 是与这个状态无关的，所以可以考虑 *j* 为定值，简化问题，这样做的时间复杂度为 $O(nmp \times \max(n,m))$，仍然超时。我们考虑从 *n* 开始倒序枚举 *i* 的过程中，定义对于当前的 *i*，转移的最大 *i'*为 *maxi*。则对于任意的 $i_1<i_2$，有 $maxi_1<maxi_2$，所以可以用单调队列进行优化，时间复杂度变为 $O(nmp)$。

1509: [NOI2003]逃学的小孩

有这样一个方法但是我不会证明，首先在树上随便选一条直径，把这条直径的两点就直接定位是 A 和 B，接下来我们枚举 C，那么当前的结果就是 $dist(A,B)+\min(dist(A,C),dist(B,C))$，取最大的一个作为答案就可以了。

1560: [JSOI2009]火星藏宝图

DP 需要一些优化。我们考虑三个点 A，B，C，其中 A 在 B 的上方，C 在 B 的右下方，定义状态 $dp[i]$表示走到第 i 个点的最大获利，v_i 表示第 i 个点物品的价值。我们可以发现从 A 走到 B 再到 C 比从 A 直接到 C 的获利要高。证明是这样的：

设 AB 距离为 d_1，BC 距离为 d_2，AC 距离为 d_3，$\angle ABC=q$，

通过 B 点的获利 $W_1=dp[A]+v_b+v_c-d_1^2-d_2^2$，

不通过 B 点的获利 $W_2=dp[A]+v_c-d_3^2$，

$W_1-W_2=v_b+d_3^2-\left(d_1^2+d_2^2\right)=v_b+d_1^2+d_2^2-2d_1d_2\cos q-\left(d_1^2+d_2^2\right)=v_b-2d_1d_2\cos q$，

因为 $v_b>0$，$d_1d_2\cos q\le 0$，所以 $W_1-W_2>0$。

因此我们考虑将所有点按照 x 坐标为第一关键字增序，y 坐标为第二关键字增序排序。那么一个点的 (x_0,y_0)，它前面的点的 x 坐标一定不大于 x_0，那么对于它来讲的合法决策一定是满足 $y\le y_0$，x 坐标最大的那个点。因此我们只需要记录 $last[i]=j$ 表示 y 坐标为 i 的点，x 坐标最大的点是第 j 个点即可。这样做的时间复杂度为 $O(nm)$。

1563: [NOI2009]诗人小 G

我们设读入的三个数分别为 n,L,P，第 i 行的长度为 $a[i]$，前 i 行的长度之和为 $sum[i]$。朴素的 DP 方程很好写：$dp[i]=\min(dp[j]+\text{abs}(sum[i]-sum[j]+i-j-1-L)^P)$ $(0<=j<i)$。但是我们发现这个方程无法用单调队列优化，当 $P\neq 2$ 的时候也无法用凸壳优化，所以猜测原问题具有决策单调性。打表发现决策确实有单调性，所以我们可以用二分区间的方式进行维护。这道题的恶心地方就是在精度会挂，能用 64 位整数的地方尽量用 64 位整数，不能用了再用 double，都用 double 则会挂精度，挂个位和十位。时间复杂度 $O(Tn\log n)$。

1571: [Usaco2009 Open]滑雪课 Ski

简单的 DP。状态 $dp[i][j]$表示在时间 i 能力值为 j 能滑的最多次数。则 $dp[i][j]=\max(dp[i-1][j]$，dp[上课][上课前的能力值]，$dp[i$ 时刻往前滑雪一次$][j]+1)$。

1572: [Usaco2009 Open]工作安排 Job

首先，我们按照结束时间进行排序，然后维护一个小根堆。对于当前的第 i 个工作，如果他的结束时间大于堆中元素的个数，那么把他的收益插入堆。否则进行判断，如果第 i 个工作的收益小于堆顶元素则忽略掉，否则将堆顶元素删除，并把第 i 个工作的收益插入堆。最后的答案是堆中所有元素的总和。

1574: [Usaco2009 Jan]地震损坏 Damage

这道题目比较有趣。我们这样想，如果一个点没有被破坏，我们设这个点为 ***v***，设与它相邻的点为 ***u***。如果 ***v*** 没有被破坏但是 ***v*** 还不能到达点 1，那么 ***u*** 一定被破坏，证明是显然的，如果 ***u*** 没有被破坏，那么 ***v*** 就可以先到 ***u***，然后从 ***u*** 到达点 1。有了这样的想法，我们把所有没有被破坏但是不能到达点 1 的点全部染色。然后从点 1 开始做 BFS，所有能宽搜到的点都是可以到达的，设个数量为 *m*。则答案就是 *n*－*m*。

1575: [Usaco2009 Jan]气象牛 Baric

简单 DP，状态 *dp*[*i*][*j*]表示选择了 *i* 头，其中最后一头是第 *j* 头，需要的最小代价，转移很显然了，时间复杂度 $O(n^3)$。

1577: [Usaco2009 Feb]庙会捷运 Fair Shuttle

这是一道贪心题目。首先我们按照所有的起点进行从小到大排序。对于第 *i* 段乘客，首先让车上所有到站的乘客都下车，并更新答案。然后用第 *i* 段乘客把车塞满，能塞多少就塞多少。如果发现塞满了还有剩余的乘客则进行判断，不断用第 *i* 段的到达位置和车上最大的到达位置比较，只要第 *i* 段的到达位置大于车上最大的到达位置，就把这个最大的踢下车，换一个第 *i* 段的乘客上车。整个过程可以用两个堆或者一棵平衡树维护。

1578: [Usaco2009 Feb]Stock Market

由于限制了答案的大小，所以可以对每天都做一次无穷背包，并把每天挣的最大值加入答案即可。

1579: [Usaco2009 Feb]Revamping Trails

分层图最短路径，我用的 Dijkstra ＋ Heap。如果 SPFA 写的好，SPFA 也能过，但有可能被卡。

1583: [Usaco2009 Mar]Moon Mooing

开两个队列，一个用来存函数 1 生成的结果，一个用来存函数 2 生成的结果。每次取两个队列队首元素较小的一个再生成两个加入队列。注意如果一个队列中的元素已经很大了，那就不要再生成了，防止类型溢出。

1584: [Usaco2009 Mar]Cleaning Up

我们预处理一个 *skip*[*i*][*j*]数据，表示 *j* 往前跳，产生了 *i* 个不同的数，最远可以跳到的位置。容易观察出，任意一段不同的数的个数不能超过根号 *n* 个，否则就不如一个一个分效果好。状态 *dp*[*i*]表示到第 *i* 个数，不和谐度的最小值，则有 $dp[i]=\min(dp[skip[j][i]-1]+j\times j)$。答案则是 *dp*[*n*]。

1585: [Usaco2009 Mar]Earthquake Damage 2

网络流啊，可我第一次建图建错了，囧。由于是与点有关的问题，我们把每个顶点拆点成 ***v*** 和 ***v***’并连一条容量为 1 的边。对于输入的每一条边<***v***,*u*>我们连边<***v***’,*u*>和<*u*’,***v***>，容量都为正无穷。对于所有没有被破坏但是无法到达点 1 的点，设这样的点为 *p*，我们连边<*p*,*T*>（注意不是<*p*’,*T*>），容量为正无穷。其中 *T* 是设置的超级汇点，而源点我们设为 1’（注意不是 1），如果不喜欢这样做的话可以连一条边<1,1’>，容量为正无穷，然后设 1 为源点。然后求这个网络的最小割就是答案了，有人说这样拆点会超时，但是我用带 GAP 优化的 SAP 求可以过，挺快的。

1589: [Usaco2008 Dec]Trick or Treat on the Farm

这道题其实有很简单的做法，但是我没有想出来。我的做法比较麻烦，代码也不短。首先我们可以发现如果一些点在一个环中，那么这些点最终计算出的答案一定是一样的。所以先用 Tarjan 求一次强连通分量。接着对于联通块建立新图并统计每个联通块节点的个数。由于缩点后的图是拓扑有序的，我们进行一次拓扑排序。然后按照拓扑序的倒序求出每个联通块的答案，那么每个点对应的答案就是它所在的联通块的答案，最后输出即可。

1590: [Usaco2008 Dec]Secret Message

把模板串插入进字母树，定义每个节点的 cnt_1 表示经过该节点的模板串个数，cnt_2 表示以该节点为结尾的模板串个数。则对于任意一个等待匹配的串，答案是路径上所有 cnt_2 的和，加上到达结尾时候的那个节点的左右儿子的 cnt_1 的和。

1592: [Usaco2008 Feb]Making the Grade 路面修整

这是个 DP，但是需要观察出一个性质，那就是一定存在一种方案，使得对于修改的每个数，都在原数列中出现过，并且结果可以最优。有了这样的一个性质，我们再开一个数组对原数列进行一次排序。我们定义 $dp[i][j]$ 表示将第 i 个数修改为排序后的第 j 个数并且保证单调不减所需要的最小代价（单调不增同理）。我们设原数列为 $a[i]$，排序后的数列为 $sa[i]$。则有 $dp[i][j]=\min(dp[i-1][k])+\text{abs}(a[i]-sa[j])$，其中 $1<=k<=i$。显然这样做的时间复杂度是 $O(n^3)$。考虑到对 k 具有连续性，我们定义 $dpmin[i][j]=\min(dp[i][k])$，其中 $1<=k<=j$，这个数组可以在求 DP 的过程中求出。则原方程变为 $dp[i][j]=dpmin[i-1][j]+\text{abs}(a[i]-sa[j])$。这样做时间复杂度就变成了 $O(n^2)$。

1593: [Usaco2008 Feb]Hotel

我们定义如果一个旅店为空，那么它的颜色是 1，如果已经有人入住，那么它的颜色是 0。算法是线段树，操作有两个，把[*L*,*R*]变成 0 或 1，询问是否存在长度为 *len* 的连续为 1 的区间，如果存在输出起点最小的一个。第一个操作不说了，打一个 *lazy* 标记就可以了。对于第二个操作，我们维护线段树每个节点的四个域，我们设当前节点的左端点为 *left*，右端点为 *right*。*maxl* 表示包含 *left* 向右的最长连续为 1 的长度，*maxr* 表示包含 *right* 向左的最长连续为 1 的长度，*maxm* 表示整个[*left*,*right*]内，最长连读为 1 的长度，*size* 表示当前节点的长度，*size*＝*right*－*left*＋1。对于要查询的区间[*L*,*R*]，我们先确定是否存在这样的长度 *len*，如果 1 号节点，也就是整棵线段树的 *maxm* 都不足 *len* 那么一定无解，反之必然有解。下面考虑有解的情况：对于一个节点，如果这个节点的 *maxm*>＝*len* 则说明有解，其中有三种情况：1.长度为 *len* 的区间完全在左子树，2.长度为 *len* 的区间完全在右子树，3.长度为 *len* 的区间横跨左右子树，也就是说这个长度为 *len* 的区间是右左子树的右端点向左延伸并上右子树的左端点向右延伸。由于要求得最小的起点位置，我们按照上面三种情况的 1,3,2 的顺序进行检查就可以了。具体实现是这样的：

```
int Query(int ro,int len) //节点序号 询问的长度
{
```

```
    Pushdown(ro); //注意要把 lazy 标记进行下传
    if (seg[ro<<1].maxm>=len) return Query(ro<<1,len);
    if (seg[ro<<1].maxr+seg[ro<<1|1].maxl>=len) return seg[ro<<1].right-seg[ro<
<1].maxr+1;
    return Query(ro<<1|1,len);
}
```

1594：[Usaco2008 Jan]猜数游戏

这道题很不错。首先我们发现如果直接求的话会带来很多困难，因为区间之间的制约关系太多，所以我们二分答案转变为判断性问题，下面讲如何判断当前的 *mid* 是否合法。我们取出 1 到 *mid* 这些区间，按照最小值为第一关键字降序排序，起点为第二关键字升序排序，终点为第三关键字升序排序。那么我们会发现这样一个事实，我们总是先处理最小值大的后处理最小值的小的。如果我们发现当前的区间已经被完全覆盖，并且覆盖的那个最小值大于当前的，那么显然是不合法的。具体实现我们可以把整个数轴建成一棵线段树，对每个节点设置一个 *cover* 的 *lazy* 标记。如果当前的区间的最小值不同于上一个区间，我们则看当前区间是否已经完全覆盖，如果是的话显然不可行。剩下的情况就是当前区间和上一个区间的值相同，我们取出所有这些最小值相同的区间，如果他们没有公共部分，那么显然不可行，因为题目说每个数都不同，否则的话我们看这个公共部分有没有被覆盖，如果有那么也不可行，如果这些都通过了，那么我们就把这些最小值相同的区间的并集求出，覆盖即可。

1596：[Usaco2008 Jan]电话网络

这道题还是有一些思维含量的，做法就是 TreeDP。定义状态 $dp[i][0],dp[i][1],dp[i][2]$分别表示在第 i 个节点建，在 i 的父节点建而 i 自身不建，在 i 的子节点建而 i 自身不建。则有转移方程：（j 表示 i 的子节点）$dp[i][0]=\min(dp[j][0,1,2])+1$，$dp[i][1]=\min(dp[j][0,2])$，$dp[i][2]=\min(dp[j][0,2])$但是第三种情况比较特殊，如果进行 min 决策时，所有的 j 都选择了 $dp[j][2]$，而无一选择 $dp[j][0]$，这样是不可以的，因为这样的话 i 无法被覆盖，所以需要进行特判。特判的原则就是如果所有选的都是 $dp[j][2]$，则将最小的一个 $dp[j][0]-dp[j][2]$加到 $dp[i][2]$上，以保证结果的可行性和最优性。对于叶子节点有 $dp[i][0]=1$，$dp[i][1]=0$，$dp[i][2]=\text{INF}$。依次表示自身建花费是 1 个，依靠父亲不花费，而由于叶子节点无子节点，所以花费为正无穷。

1597：[Usaco2008 Mar]土地购买

这道题应该是一个动态规划模型，但是需要转换。我们观察题意，购买一组土地的费用等于最大的长乘最大的宽，我们设这样的两块地，第一块长为 a 宽为 b，第二块长为 c 宽为 d，如果 $a>=b$ 且 $c>=d$，那么可以认为第二块属于“赠送土地”，因为我们只要在买第一块的那组，附带上第二块就可以了，这不影响费用。于是我们想到按照所有土地的长进行递减排序，长一样的按宽递减排序，我们用这种方式去掉所有的“赠送土地”。我们先把排序后的第一块土地加入到我们需要决策的队伍中，此后，从第 2 块到第 n 块中，依次与最后一块加入的土地进行比较。如果当前一块的长比最后一块长且当前一块的宽比最后一块宽，则加如，否则不加入，因为按照我们的排序原则这样做不会把该加入的丢掉，也不会把不该加入的加入。接下来显然所有土地的长为递减，所有土地的宽为递增。我们设 $ch[i]$表示第 i 块土地的长，$ku[i]$表示第 i 块土地的宽，$dp[i]$表示购买完毕第 i 块土地，所需的费用的最小值。则有 $dp[i]=\min(dp[j]+ch[j+1]\times ku[i])$，其中 $0<=j<i$。显然这样做的时间复杂度为 $O(n^2)$，会超时。我们进行优化，首先证明这个东西的决策单调性，也就是说设 $dp[i]$的最小值从 j 决策得到，$dp[i']$的最小值从 j' 决策得到，则有当且仅当 $i<i'$时有 $j<=j'$，所以我们只需证明这个决策满足四边形不等式即可。我们设 $w[i,j]=ch[i]\times ku[j]$。也就是说需要证明 $w[i,j]+w[i+1,j+1]<=w[i+1,j]+w[i,j+1]$。显然有 $ch[i]>=ch[i+1]$，$ku[j]<=ku[j+1]$，我们设 $ch[i]=ch[i]+x$，$ku[j]=ku[j+1]-y$，其中 $x,y>=0$。令 $A=w[i,j]+w[i+1,j+1]$，$B=w[i$

＋1,*j*]＋ ***w***[*i*,*j*＋1]。则有：

$$\mathbf{Q}\ A = ch[i]*ku[j]+ch[i+1]*ku[j+1]=ch[i]*ku[j]+(ch[i]-x)*(ku[j]+y)=2*ch[i]*ku[j]-x*ku[j]+y*ch[i]-xy$$

$$\mathbf{Q}\ B = ch[i+1]*ku[j]+ch[i]*ku[j+1]=(ch[i]-x)*ku[j]+ch[i]*(ku[j]+y)=2*ch[i]*ku[j]-x*ku[j]+y*ch[i]$$

$$\therefore A-B=-xy\le 0 \quad \therefore A\le B$$

所以原方程满足四边形不等式，可以用决策单调进行优化。时间复杂度为 *O*(*n*log*n*)，或者直接斜率优化，时间复杂度 *O*(*n*)。

1598: [Usaco2008 Mar]牛跑步

这道题我们可以用 *n* 棵平衡树来维护到达每个点的第 1,2，……，*k* 短路。我们按照景点倒序的顺序进行处理，对于当前的景点 *i*，我们用 *i* 所在的平衡树的每一个距离进行处理，分别处理 *i* 可以到达的点 *j*，用这个距离加上 *i* 和 *j* 的距离对 *j* 进行松弛，如果 *j* 所在的平衡树中元素个数不超过 *k* 个，则插入平衡树，否则判断是否将平衡树中的最大值删除，并替换当前的距离即可，平衡树可以用 STL 中是 *set* 代替。

1599: [Usaco2008 Oct]笨重的石子

这数据范围……三次方枚举就过了。考虑数据范围更大的情况：*dp*[*i*][*j*]前 *i* 个骰子，组成和为 *j* 的方案数，*dp*[*i*][*j*]＝Σ*dp*[*i*－1][*k*]，设第 *i* 个骰子的点数为 *p*，其中 1<＝*j*－*k*<＝*p*

1602: [Usaco2008 Oct]牧场行走

随意一种方法求出所有询问的LCA，设*dist*[*i*]表示点1到点*i*的距离，则*ans*＝*dist*[*i*]＋*dist*[*j*]－2×*dist*[lca(*i*,*j*)]。数据范围很小，预处理任意两点距离应该也可以。

1605: [Usaco2008 Open]Crisis on the Farm

取第 1 座塔为参考系，其余输入信息对应调整，预处理 *map*[*i*][*j*]为第 1 座塔移动到 *i*，*j* 可以拯救的数量。*dp*[*k*][*i*][*j*]＝max(*dp*[*k*－1][*i*±1][*j*±1])，*dp*[0][0][0]＝0 记得维护字典序最小以及数组的负下标。

1611: [Usaco2008 Feb]Meteor Shower 流星雨

首先用读入的数据处理每个格子受到攻击的最早时间，如果不会受到攻击就设为无穷大。宽搜一遍，注意去过的格子不要再去，否则队列开不下，如果到达了一个最早受攻击时间为正无穷的格子，那么此时的时间就是答案。

1615: [Usaco2008 Mar]The Loathesome Hay Baler

首先确定开始和结束的齿轮编号，然后将所有齿轮的能否连通整理成一个二维 bool 数组，从开始齿轮开始 dfs，记录每个齿轮的 *father*，最后按照 *father* 序列及题目的计算方法算一下就可以了。

1619: [Usaco2008 Nov]Guarding the Farm

按照高度排序以后，做 floodfill，注意如果相邻的格子和当前格子的高度一样也要覆盖。

1620: [Usaco2008 Nov]Time Management

按结束时间从大到小排序，然后做一遍可以了。有一个坑人的地方就是这道题有一个测试点是无解的，但是题目没有说无解的情况如何处理，输出－1 就对了。

1623: [Usaco2008 Open]Cow Cars

根据输入数据求出每头牛的前面可以有多少头牛设为 $a[i]$，按照 $a[i]$递增的顺序排序，贪心放即可，如果能放就放，可以保证最后答案最优。

1624: [Usaco2008 Open] Clear And Present Danger

根据输入信息，求出任意两点的最短路长度，用 floyd 即可。然后根据输入的那个序列，设这个序列的长度为 m，$map[i][j]$表示点 i 到点 j 的距离，$seq[i]$表示序列中的第 i 个数，则 $ans=\Sigma map[seq[i-1]][seq[i]]$，$i\in[2,m]$。

1628: [Usaco2007 Demo]City skyline

首先矩形的宽度和题目给的总宽度对题目答案没有影响，直接忽略，然后维护一个单调栈，如果当前高度大于，则进栈，等于说明没用，直接忽视，小于开始弹，并更新答案，弹到大于为止，最后栈里可能剩东西，所以再加入一个高度为 0 的矩形即可解决这个问题。证明是显然的，画画图就好了。

1629: [Usaco2007 Demo]Cow Acrobats

按照每只牛的 w 与 s 的和为关键字排序，然后扫一遍。可以考虑调整法证明。

1630: [Usaco2007 Demo]Ant Counting

比较简单的 DP，$dp[i][j]$前 i 个家族去了 j 个蚂蚁的方案数 $dp[i][j]=\Sigma dp[i-1][j-k]$ $k\in[0,a[i]]$ $j-k$ 非负，$a[i]$是第 i 个家庭蚂蚁的总数，对于那个求和可以用前缀和解决，加上滚动数组就可以过了。

1632: [Usaco2007 Feb]Lilypad Pond

在 SPFA 的时候改造一下就行了，定义 $ans[i][0,1,2]$分别表示最少添加的荷叶数，在最少添加荷叶的情况下最少移动步数，在最少添加荷叶最少移动步数情况下的方案数，然后把 SPFA 的松弛条件修改一下就可以了。

1633: [Usaco2007 Feb]The Cow Lexicon

$dp[i]$表示原串到第 i 个字符，最长保留的长度，答案则是 $n-dp[n]$，转移是 $dp[i]=\max(dp[j]+\boldsymbol{w}[k].length())$ 从第 $j+1$ 个字符到第 i 个字符，可以与第 k 个小串进行匹配，这里的匹配是指去掉一些字符后与小串相同。

1634: [Usaco2007 Jan]Protecting the Flowers

这道题目挺有意思的，用贪心的方法对数据进行排序，我们设第 i 头牛的时间为 t_i，破坏度为 d_i，我们只需按 t_i/d_i 为关键字进行排序就可以了，下面给出证明：
我们令 $j=i+1$，假定我们已经安排出了一套最优方案，考虑对于最优方案时相邻的两头牛：我们设第 $j+1$ 头牛到第 n 头牛的破坏度总和为 S，则有：

$$t_i\times(d_j+S)+t_j\times S\le t_j\times(d_i+S)+t_i\times S$$
$$\Leftrightarrow t_i\times d_j+(t_i+t_j)\times S\le t_j\times d_i+(t_j+t_i)\times S$$
$$\Leftrightarrow t_i\times d_j\le t_j\times d_i\Leftrightarrow\frac{t_i}{d_i}\le\frac{t_j}{d_j}$$

原命题得证，注意答案可能需要用 64 位整数才能存下。

1635: [Usaco2007 Jan]Tallest Cow

这道题也是贪心就可以了，首先设所有的高度都为 0，对于读入的一个区间$[L,R]$，我们只需要将$[L+1,R-1]$内所有的数都−1 就可以了，注意区间可能有重复的，用一个 map 就可以解决。最后再考虑给定的那个身高，按照那个身高进行调整一次就可以了。

1637: [Usaco2007 Mar]Balanced Lineup

首先把颜色是 0 的颜色换成−1，然后按位置从小到大排序，记 $sum[i]$表示前 i 个的颜色总和，对于任意的位置 a，b，其中 $sum[a]=sum[b]$，要求最大化 $pos[a]-pos[b+1]$，记 $last[i]$表示总和为 i 个最靠前的那个，如果不存在，则 $last[i]=-1$，然后乱搞就可以了。

1638: [Usaco2007 Mar]Cow Traffic

这道题题意理解了好半天，意思是给定一个有向无环图，对于起点到终点的很多条路径中，对于一条边，路径包含这条边一些次，求最多的包含次数的那条边的次数。由于边都是编号小的连向编号大的，所以无须进行拓扑排序，1，2……n 就是原图的一个拓扑序列。考虑一条边：由 a 指向 b，那么答案就是起点到 a 的路径数目×b 到终点的路径数目。然后就很好办了……注意空间不要爆……64MB T_T……

1639: [Usaco2007 Mar]Monthly Expense

二分最终答案以后转变为判断性问题，$l=0$，$r=10^9$，贪心放即可，如果可行 $r=mid-1$ 否则 $l=mid+1$ 最后 l 就是答案。

1640: [Usaco2007 Nov]Best Cow Line

后缀数组，原串关于结尾对称翻转复制一次，求出 *rank* 数组，开一个 *l* 和一个 *r*，每次选 *l* 往右和 *r* 往左对应的 *rank* 小的加入答案就可以了。

1641: [Usaco2007 Nov]Cow Hurdles

dp[*i*][*j*]表示 *i* 到 *j* 的答案，首先令 *dp*[*i*][*j*]等于无穷大，*dp*[*i*][*i*]＝0，用读入的数据更新一次，然后用类似 floyd 的思想，*dp*[*i*][*j*]＝min(*dp*[*i*][*j*],max(*dp*[*i*][*k*],*dp*[*k*][*j*]))。然后再将所有 *dp*[*i*][*j*]等于无穷大的变成－1，读入询问，依次回答就可以了。

1642: [Usaco2007 Nov]Milking Time

dp[*i*]表示第 *i* 个时间段必须做，能取得的最大收益。按开始时间排序，n^2 做一遍就可以了。答案是 max(*dp*[*i*])。

1643: [Usaco2007 Oct]Bessie's Secret Pasture

很简单的 DP，*dp*[*i*][*j*]，前 *i* 块，和为 *j* 的方案数，dp[*i*][*j*]＝Σdp[*i*－1][*j*－*k*]，*k* 为完全平方数。

1644: [Usaco2007 Oct]Obstacle Course

记忆化搜索或者是宽搜，可我写的记忆化搜索始终没有调出来，最后就写的宽搜，注意如果转弯的时候转 180°比如说原来是向左，转成向右，那么这也算转一个弯。

1645: [Usaco2007 Open]City Horizon

由于所有的矩形的下面那条线都是对齐的，对于给出的 *n* 个矩形，我们首先按 *x* 坐标进行排序。然后维护一个大根堆，*n* 个矩形共可形成 2×*n* 条线，我们从左到右逐一扫面。对于第 *i* 条线，我们首先更新答案，设第 *i* 条线的横坐标为 *x*[*i*]，我们将答案加上(*x*[*i*]－*x*[*i*－1])×*maxh*，其中 *maxh* 是堆中所有线段的最高的那条线段的高度，这么做显然是正确的。如果第 *i* 条线是一个矩形的左边界，则将其插入堆，如果是一个矩形的右边界，则删除堆中这个矩形的左边界。时间复杂度为 *O*(*n*log*n*)。

1646: [Usaco2007 Open]Catch That Cow

做一遍 BFS 就可以了，每次把＋1，－1，×2 的三个数中没有出现的加入队列，可以加一个剪枝，我写的是如果加入的数大于等于 20 万或者小于等于－10 万就不要加入了，然后就可以过了。

1647: [Usaco2007 Open]Fliptile

这道题也是枚举就可以了，显然对于任意一个格子操作 2 次等于没操作，所以任意格子只能操作 1 次或者 0 次。虽然是一个 *N*×*M* 的格子，但是观察可以发现，如果第 1 行的操作情况一旦确定下来，那么第 1 行为 1 的格子，只能通过第 2 行的操作变回 0，接着第 2 行为 1 的，只能通过第 3 行的操作变回 0，以此类推。如果操作到最后仍然有 1 的格子，那么第 1 行的操作序列就是不合法的。我们用二进制进行枚举，这样可以保证是按照字典

序进行枚举的。共 2^m 种情况，如果当前的操作步数小于之前的最优操作步数，那么更新答案，否则保留原答案，因为要求字典序最小的方案，时间复杂度为 $O(nm\times 2^m)$。

1648: [Usaco2006 Dec]Cow Picnic

用 $O(n^2)$时间预处理任意两点 i，j 是否可达，然后判断就可以了。

1649: [Usaco2006 Dec]Cow Roller Coaster

按结束点排序，然后就变成了 01 背包问题，注意数据中存在无解的情况，需要输出－1。

1650: [Usaco2006 Dec]River Hopscotch

把输入的坐标从小到大排序，然后进行二分答案，$l=1$，$r=len$，$mid=(l+r)/2$。如果 mid 可行则 $l=mid+1$，否则 $r=mid-1$，最后的 r 就是答案。

1652: [Usaco2006 Feb]Treats for the Cows

这道题用 n^2 的 DP 就可以过了，状态是：$dp[i][j]$表示从左开始取走 i 个，从右开始取走 j 个能获得的最大收益。则有：$dp[i][j]=\max(dp[i-1][j]+a[i]\times(i+j)$，$dp[i][j-1]+a[n-j+1]\times(i+j))$，$dp[0][0]=0$。

1653: [Usaco2006 Feb]Backward Digit Sums

银组题都是神数据范围啊，$n!$ 枚举就行了，注意要按字典序枚举，找到一组解就可以停止了。说一个小优化，对于 n 个数，最后变成了 1 个数，其实并不需要按题目说的那样计算 $n-1$ 轮，观察可以发现，如果初始时第 i 个数为 $a[i]$，我们设杨辉三角第 i 列第 j 行为 $dp[i][j]$，那么最后的计算结果就为 $\Sigma a[i]\times dp[n][i]$，可以让程序跑的快一些。当然，如果数据范围更大一些，需要用极大极小剪枝，对称性剪枝等，但是显然此题不需要。

1654: [Usaco2006 Jan]The Cow Prom

先用 Tarjan 求一遍强连通分量，答案就是所有强连通分量中点数大于 1 的强连通分量的个数。

1655: [Usaco2006 Jan] Dollar Dayz

K 为外层循环枚举 i，N 为内层循环枚举 j，$dp[j]+=dp[j-i]$，需要写个高精度。

1658: [Usaco2006 Mar]Water Slides

题目意思就是问至少花多少代价可以把原图变成一个欧拉图，首先统计每个点的出度减去入度。把正负分成两组，然后用 x 的值排序，排序以后只要按照 x 大的配 x 大的原则进行贪心就可以了，证明略了。

1659: [Usaco2006 Mar]Lights Out

囧，这道题没有数据＝ ＝定于状态 $dp[i][j]$表示到达牛的爪子的第一个脚趾头在 i 位置，并且 i 到 $i+T-1$ 这一段用二进制表示位 j 的时候，最少量多少个。$dp[i][j]=\min(dp[i-1][k]-1/-0)$。其中状态 k 二进制右移一位与状态 j 除了最后一位都相同。

1660: [Usaco2006 Nov]Bad Hair Day

最开始想对高度离散化后用线段树维护，发现 10 个点的时间一共才 2s……所以，维护一个元素单调递减的单调栈，对于第 i 头牛，设栈中元素有 a 个，则第 i 头牛可以被 a 头牛看到，更新答案，此时弹出直到栈顶元素大于第 i 头牛的身高，或者栈空。对于第 i 头牛，最多进栈一次，出栈一次。时间复杂度为 $O(n)$。

1661: [Usaco2006 Nov]Big Square

枚举正方形一条边的两个顶点，然后算出另两个顶点，如果这四个点有三个或四个是 J，且任意一个都不是 B，那么更新答案。

1662: [Usaco2006 Nov]Round Numbers

这题可以用 DP 解决，但是边界条件什么的很麻烦，所以我们用打表解决。数据范围是 2×10^9，每隔 10^6 打一个就可以，然后最多只需要算 10^6 个，剩下的用暴力解决，打表很快的，几分钟就能打出来。

1663: [Usaco2006 Open] The County Fair

按照开始时间排序然后进行 DP 就可以了，有一个问题需要注意，我们设每个东西的开始时间是 T_i，输入的二维数组为 $a[i][j]$，边界条件不是 $dp[1]=1$，而是 $dp[i]=1$，其中 $T_i>=a[1][i]$。

1664: [Usaco2006 Open]County Fair Events

这道题有两种做法，先说一种比较好想的，我们按照开始点进行排序，设 $dp[i]$表示第 i 个节目必看，可以看的最多节目，先令 $dp[i]=1$，然后显然有 $dp[i]=\max(dp[j]+1)$，$j<i$ 且第 j 个节目的结束时间小于等于第 i 个节目的开始时间，边界条件 $dp[1]=1$，$ans=\max(dp[i])$，这样做的时间复杂度是 $O(n^2)$。第二种做法比较巧妙，题目说的每个节目的开始时间和持续时间小于等于 10^5，我们要充分利用这个条件。设 $start[i]$表示第 i 分钟结束的节目的最晚开始时间，这个可以用输入的数据处理出来，如果不存在第 i 分钟结束的节目，则令 $start[i]=0$。再设 $dp[i]$表示看到第 i 分钟，能看的最多节目数，显然有：$dp[i]=dp[i-1]$，$dp[i]=\max(dp[i],dp[start[i]]+1)$，其中 $start[i]\neq0$。我们令 $T=200000$，这样做的时间复杂度是 $O(T)$，$ans=dp[T]$。

1665: [Usaco2006 Open]The Climbing Wall

这是一道最短路问题，首先我们处理对于不同的两点 i，j 是否可以互相到达，为了减少预处理的时间，我们按照 x 排序，顺序枚举 i，从 $i+1$ 开始枚举 j，如果 i 和 j 的横坐标已经相差 1m 以上，那么此时 j 往后的都不用

继续枚举，因为后面的 j 都无法与 i 连边。设置起点 S，终点 T，S 和 y 小于等于 1000 的连边，总高度减去 y 小于等于 1000 的和 T 连边。求一遍以 S 为源的单源最短路，S 和 T 的最短距离减 1 就是答案。

1667: [Usaco2006 Oct]Cows on Skates 滑旱冰的奶牛

BFS 一遍，把路径记录下来就可以了。我交了一次，发现是 WA，结果是 BZOJ 上没有这题的输出数据，只要什么都不输出就可以过了，囧。

1668: [Usaco2006 Oct]Cow Pie Treasures 馅饼里的财富

就是个数塔问题，$dp[i][j]$，第 i 行第 j 列能取到的最大和。边界条件，$dp[1][1]=a[1][1]$，$dp[i][1]=$负无穷($i\neq1$)。

1670: [Usaco2006 Oct]Building the Moat 护城河的挖掘

题目要求用一圈线，把所有的点都包上，或者点在线上。直接求二维凸包就行了。

1671: [Usaco2005 Dec]Knights of Ni

这道题我们可以从 2 开始进行宽搜，求出 2 到达每个格子的最短距离。然后从 3 开始再进行一次宽搜，求出到达每个格子的最短距离，我们设第一次求出的距离为 $dist[0][i][j]$，第二次求出的距离 $dist[1][i][j]$，那么答案就是 $\min(dist[0][i][j]+dist[1][i][j])$，其中 $map[i][j]=4$。

1672: [Usaco2005 Dec]Cleaning Shifts

这道题是一个 DP。但是需要优化。我们设需要维护的大时间段是$[S,T]$，首先我们把每个时间段按照结束时间进行排序，先处理无解的情况，如果结束时间最晚的那一个段的结束时间小于 T，显然无解。如果大于等于 T，还需要继续判断，我们将最后一段开始往左连，连成一条包含 T 的线段的最长连续线段，我们设连成的这条线段的左端点的最小值为 P，如果 $P>S$，那么显然无解，剩下的情况就都是有解的了。对于有解的情况，我们设 $dp[i]$ 表示第 i 段必须选取，到第 i 段的结束时间位置，最小的花费，设第 i 段的起始时间为 $be[i]$，结束时间为 $en[i]$，第 i 段的花费为 $p[i]$。则有：$dp[i]=\min(dp[j]+p[i])$，其中 $en[j]>=be[i]-1$。答案则是 $\min(dp[i])$，其中 $en[i]>=T$。这样做的时间复杂度是 $O(n^2)$，如果递减枚举 j，在不符合条件的时候就停止的话，应付 USACO 的数据的可以的。我们考虑一下优化情况，首先对方程进行化简，$p[i]$是一个常量，可以移到括号外，则有 $dp[i]=\min(dp[j])+p[i]$，其中 $en[j]>=be[i]-1$。我们发现对于满足条件的 j 总是连续的一段，因为我们是按照结束时间进行排序的。这一段的终点可以认为是 $i-1$（或者不存在）。我们只需要找到这个起点，这个起点可以用二分查找在 $O(\log n)$的时间内找到，设这个起点位置为 j。这样问题就变成了 $dp[i]=\min(dp[k])+p[i]$，其中 $j<=k<=i$。对于一段数最小值的问题可以使用线段树解决，对 i 决策完毕时，把 $dp[i]$插入线段树就可以了。这样做的时间复杂度为 $O(n\log n)$。

1673: [Usaco2005 Dec]Scales

注意到第 i 个的体积大于等于第 $i-1$ 个和第 $i-2$ 个的和，可以发现物体的总个数不会太多，最多也就是四五十个，所以考虑搜索。说几个剪枝，首先按照体积排序，把大于限制的全部扔掉，然后贪心算一组解，为了以

后剪枝方便。搜索的时候，按照体积从大到小搜比从小到大搜快，这是显然的。如果当前的总体积大于限制，剪枝。如果当前的总体积，加上剩下所有物品的总体积仍然不比答案更优，剪枝。加上这些剪枝可以全部 0ms 解决。

1674: [Usaco2005]Part Acquisition

宽搜或者求最短路，最短路＋1 就是答案，注意无解输出－1。

1675: [Usaco2005 Feb]Rigging the Bovine Election

这道题爆搜就可以了，最多也就是 2^{25} 种情况，但是如果常数写的不好就会 T。加一个剪枝，如果已经有了 7 个 1 就可以去 check 了，这样就可以过了，check 的时候对所有的 1 进行 floodfill，如果连通块不是 1 个那说明不行。

1676: [Usaco2005 Feb]Feed Accounting

从 D 往回枚举就行了。

1680: [Usaco2005 Mar]Yogurt factory

简单的 DP，$dp[i]$表示第 i 天的最低单价是多少，我们设每天的增量为 s，第 i 天的输入单价为 $a[i]$，显然有 $dp[i]=\min(dp[i-1]+s,a[i])$，边界条件 $dp[1]=a[1]$。

1681: [Usaco2005 Mar]Checking an Alibi

以 1 为源，求一次单源最短路。写最好写的就行，bellman_ford 吧。

1682: [Usaco2005 Mar]Out of Hay

直觉告诉我，用 kruskal 求最小生成树，加入的最后一条边就是答案，这样做是正确的，证明……我也不知道为啥。

1683: [Usaco2005 Nov]City skyline

经典问题，单调栈维护。

1685: [Usaco2005 Oct]Allowance

不知道这题在考什么，照着模拟一遍一遍找就过了。

1687: [Usaco2005 Open]Navigating the City

这题是真心无聊啊，首先输入的信息就极其恶心，需要处理成正常的 bool 数组或者地图什么的。然后大家都懂的，宽搜一遍就可以了。输出的时候注意一下，有点麻烦。

1688: [Usaco2005 Open]Disease Manangement

这题也是暴力就可以过的＝ ＝最多 15 种病，也就是 2^{15}＝32768 种情况，时间复杂度 $O(n\times 2^D)$ ，D 为输入数据中共有多少种疾病。

1689: [Usaco2005 Open] Muddy roads

贪心就可以了，首先按读入数据的起点进行排序。然后扫描一遍，如果需要放就放，当然如果是空地显然不用放。这样做可以保证最终答案是最优的。

1690: [Usaco2007 Dec]奶牛的旅行

这是一道 01 分数规划问题。显然答案具有单调性，所以我们进行二分答案，现在就变成了如何判断当前的 *mid* 是否可行。可行的条件显然是Σ点权/Σ边权>＝*mid*。也就是说Σ点权－*mid*×Σ边权>＝0。我们把所有的点权都加到边权上。定义当前边的<*u*,*v*>的边权为 *mid*×边权－***v*** 的点权。则问题转换成Σ新边权<＝0。由于精度要求不高，我们可以二分到小数点后 4 位，对新的边用 SPFA 判负环就可以了。时间复杂度为 *O*(二分次数×*nm*)。

1694: [Usaco2007 Demo]Grazing on the Run

很难想的 DP。做法是 *dp*[*i*][*j*][0]表示算完闭区间[*i*,*j*]并且最后站在 *i* 位置上所需要的最少时间，*dp*[*i*][*j*][1]则是站在 *j* 位置。那么我们可以用 *dp*[*i*][*j*][0]，*dp*[*i*][*j*][1]来更新 *dp*[*i*][*j*＋1][1]，*dp*[*i*－1][*j*][0]。但这里就是本题的最大难点。以 *dp*[*i*][*j*＋1][1]为例，我们设总长度问 *n*，那么抛去[*i*,*j*]这段的长度 *l*＝*n*－(*j*－*i*＋1)。则有 *dp*[*i*][*j*＋1][1]＝min(*dp*[*i*][*j*][0]＋*l*×(*a*[*j*＋1]－*a*[*i*]),*dp*[*i*][*j*][1]＋*l*×(*a*[*j*＋1]－*a*[*j*])。只所以要乘以那个长度是因为这段 *a*[*j*＋1]－*a*[*i*]的这段损耗将不止被计算一次，我们需要把它对后面的影响都考虑在内。那么最终的答案就是 min(*dp*[1][*n*][0..1])。

1695: [Usaco2007 Demo]Walk the Talk

我们发现词典中单词的长度最多为 4，共有 2265 个单词。所以我们可以对每一个单词单独处理，假定我们处理第 *P* 个单词，定义状态 *dp*[*k*][*i*][*j*]表示处理到第 *k* 个字母，在格子第 *i* 行，第 *j* 列上的方案总数。则有 *dp*[*k*][*i*][*j*]＝Σ*dp*[*k*－1][*i*'][*j*'](*i*'>＝*i* 且 *j*'<＝*j* 且 *i*,*i*',*j*,*j*'不同时相等)，为了加速运算，求和过程可以用一个前缀和优化。对于每个单词的答案为Σ*dp*[*len*][*i*][*j*](1<＝*i*<＝*n* 且 1<＝*j*<＝*m*)，其中 *len* 为第 *P* 个单词的长度。

1697: [Usaco2007 Feb]Cow Sorting 牛排序

裸的置换群。

1704: [Usaco2007 Mar]Face The Right Way

这道题我们找到从左往右的第一个 1，那么一定应该从这个 1 开始往右取反 k 个，我们可以令 $dp[i]=a[i]-a[i-1]$，那么只需要做一次线性扫面就可以了。时间复杂度为 $O(n^2)$。

1706: [usaco2007 Nov]relays

这道题可以巧妙的用矩阵乘法解决，具体参见 2008 年国家集训队论文《矩阵乘法》。

1712: [Usaco2007 China]Summing Sums

找到规律以后用矩阵乘法做就可以了。

1716: [Usaco2006 Dec]The Fewest Coins

很容易想到这是一个有限背包和无限背包的组合问题。我们设所有硬币的最大面值为 W。那么对于找零的那部分我们需要做无穷背包，其实这个枚举上限只要到 W^2 就可以了。证明是这样的，如果找零超过 W^2 的钱，那也就是说明找零的硬币最少为 $W+1$ 个，那么这些硬币一定包含了全部种类的硬币，所以这样的方案一定不是最优的，因为可以在给钱的时候少给一些，从而让找零少一些。对于有限背包的一部分用单调队列优化就可以了。

1718: [Usaco2006 Jan] Redundant Paths

先求一遍桥，然后把所有桥删掉，对剩下的残图求一下连通块，这样就得到了一个一个的双连通分量。然后把每个双连通分量缩成一个点，把桥全加回来，现在就变成了一棵树，原问题就变成了一棵树最少填多少条边能变成一个双连通图。我们设树中有 a 个度为 1 的点，当 $a=1$ 时，答案是 0，否则是 $(a+1)/2$，为什么是这样我也不知道。

1720: [Usaco2006 Jan]Corral the Cows

由于点数非常少，但是坐标却很大，我们首先对横纵坐标离散化成 1－500 之间的数值，记 $s[i][j]$ 表示以 i，j 为右下角，有多少块草地。（i，j 都是离散化以后的结果）。然后进行二分，二分的时候枚举最右下角是哪一块就可以了。

1721: [Usaco2006 Mar]Ski Lift

DP 就可以了。设 n 为杆子的总数，m 为相邻两个最大的距离，$dp[i]$ 为到达第 i 个杆子需要的最小值。$dp[i]=\min(dp[j]+1)$，$1<=j<=i-m$，其中第 i 个杆子与第 $j+1$ 到 $i-1$ 个杆子分别连线，设组成斜率的最小值为 k，则必须满足第 i 个和第 j 个连线的斜率<＝k，维护一下那个斜率就可以了。答案是 $dp[n]$。

1727: [Usaco2006 Open]The Milk Queue

比较难想的排序。我们设每头牛的两个参数分别为 *a*，*b*。排序的原则是把所有 *a*<*b* 的牛排在前面，*a* 越小排名越靠前，剩下的牛把 *b* 大的排在前面，然后扫描一遍。

1733: [Usaco2005 feb]Secret Milking Machine

首先可以想到二分，接着剩下的唯一问题就是如何检验 *mid* 是否可行。我们的目标是找到 *T* 条从 1 到 *n* 的不重复路径，可以用网络流解决。把所有边权小于等于 *mid* 的边全部加入网络，看最大流是否大于等于 *T*。如果是则可行，不是，则不可行。那么这就有另一个问题了，对于有向边，在加入网络的时候，会加入两条，是都有可能被流的，然而题目要求只能流一次。其实这样做了算出的答案也是正确的，比如说有 *a*，*b*，*c*，*d*，*e*，*f* 六个点。第一条路径是 *a*－>*b*－>*c*－>*d*，第二条路径是 *e*－>*c*－>*b*－>*f*。虽然这样是不符合要求的，但我们可以转换一下。从第一条路径我们可以看出 *c* 可以到达 *d*，从第二条路径我们可以看出 *b* 可以到达 *f*。所以把第一条路径改写成 *a*－>*b*－>*f*，把第二条路径改写成 *e*－>*c*－>*d*，这样问题就完美解决了。

1734: [Usaco2005 feb]Aggressive cows

金组也有这种题＝ ＝。二分答案，剩下你懂的^_^。

1735: [Usaco2005 jan]Muddy Fields

这道题最开始想错了，以为只要把联通块求出来，每一个联通块内都做一次二分图匹配，然后相加就行。画画图可以发现这是错误的。正确的做法是，对于每个星号，求出成行的有多少组，记此时有 *a* 组，成列的有多少组，记此时有 *b* 组。下面建立二分图，左边为 *a* 组的所有序号，右边为 *b* 组的所有序号。我们设原图为 *map*[*i*][*j*]，如果 *map*[*i*][*j*]为星号，我们就在 *map*[*i*][*j*]在 *a* 组中的编号向 *map*[*i*][*j*]在 *b* 组中的编号连一条边。显然求最小路径覆盖即可，也就是最大匹配。

1736: [Usaco2005 jan]The Wedding Juicer

贪心，用优先队列（堆）进行优化。首先我们把四周的点放进堆，直到堆不空为止我们不断做这样一个事情：取出堆中高度最小的点 *x*，更新与 *x* 相邻的四个点，如果相邻的点的高度小于 *x*，则把这个高度改为 *x*，并将答案加上这个差值，如果这个点没有插入过，则插入这个点。

1737: [Usaco2005 jan]Naptime

我们先考虑一条链的情况。*dp*[*i*][*j*][0,1]表示时间 *i*，睡了 *j* 个单位，第 *i* 个时间不睡或者睡获得的最大恢复值。则有：*dp*[*i*][*j*][0]＝max(*dp*[*i*－1][*j*][0],*dp*[*i*－1][*j*][1])，*dp*[*i*][*j*][1]＝max(*dp*[*i*－1][*j*－1][0],*dp*[*i*－1][*j*－1][1]＋*a*[*i*])，答案取 max(*dp*[*n*][*m*][0],*dp*[*n*][*m*][1])。考虑环的问题，首先令 *dp*[1][1][1]＝*a*[1]。然后用链的方法推一遍，这次答案要取 *dp*[*n*][*m*][1]。取两次答案中的较大值作为最终答案即可。空间卡的比较严，要滚动数组。

1738: [Usaco2005 mar]Ombrophobic Bovines

首先用 $O(n^3)$的时间预处理任意两点间的最短路。然后进行二分答案，对于当前的 *mid* 使用网络流检验。具体做法是把每个点拆成 *i* 和 *i*’。*S* 与 *i* 连容量为牛个数的流量，*i*’与 *T* 连容量为当前牛棚最多容纳个数的流量，对

于两点<*i*,*j*>，若最短路小于等于 *mid*，则由 *i* 向 *j*’连容量为无穷大的边。如果最大流和牛的总数相等则可行。

1739: [Usaco2005 mar]Space Elevator

我们设每种物品的所能放的最高位置为 *lim*，那么显然有一种最优方案使得按照 *lim* 不减的顺序放置，所以我们按照 *lim* 进行排序，然后 DP 一遍就可以了，状态 *dp*[*i*][*j*]前 *i* 个物品能否形成高度 *j*，转移就很显然了。

1740: [Usaco2005 mar]Yogurt factory

跟 1680 是一个题，双倍经验。

1741: [Usaco2005 nov]Asteroids

将行设为点集 *Xi*，列设为点集 *Yi*，如果 *map*[*i*][*j*]可以放，那么就连边 *Xi* 和 *Yj*，流量是 1，设置附加源汇 *S* 和 *T*，*S* 向所有 *X* 里的点连流量为 1 的边，所有 *Y* 里的点向 *T* 连流量为 1 的边，显然最小割就是答案。当然，可以用二分图最小路径覆盖来做。

1742: [Usaco2005 nov]Grazing on the Run

参加 1694 的题解，两个题目完全相同。

1743: [Usaco2005 nov]Walk the Talk

跟 1695 是一个题。

1744: [Usaco2005 oct]Skiing

这道题有意思的一个地方就是无论走怎样的路线，到达点(*i*,*j*)的速度都是定值。我们假设这一路经过的点分别为 a_1，a_2，a_3，……，a_k。我们设初速度为 $\boldsymbol{v}_0$，到达 a_1 时的速度 $\boldsymbol{v}_1=\boldsymbol{v}_0\times 2^{-a1}$，到达 a_2 时候的速度为 $\boldsymbol{v}_2=\boldsymbol{v}_1\times 2^{a1-a2}=\boldsymbol{v}_0\times 2^{-a2}$……。我们设读入的表格为 *map*[*i*][*j*]，则到达点(*i*,*j*)的速度为 $\boldsymbol{v}(i,j)=2^{map[1][1]-map[i][j]}$。这样就可以建立最短路模型，如果点<*a*,*b*>，<*c*,*d*>相连，则在这两点间建距离为 $\boldsymbol{v}(a,b)$的无向边。以(1,1)为源点求单源最短路，则到达点(*n*,*m*)的最短路径除以初速度就是答案。注意一点，有一个数据中 $n=m=1$，也就是说整个图只有一个点，用循环队列会悲剧，需要特判一下。

1745: [Usaco2005 oct]Flying Right

贪心就可以了，我们可以用两个堆或一个平衡树来维护。其实去和回的两个过程是相同的，我们以去为例。首先按起点进行排序，然后去处理那些起点比终点小的，因为剩下的都是回来的，这里不需要考虑。对于当前的一段时间，如果终点小于平衡树中的最大终点，就替换一下，如果有到站的，那么更新答案。

1747: [Usaco2005 open]Expedition

这道题一个贪心，能走则走。如果不能走了，记当前的位置为 *pos*。那么我们在加油站位置为 1 到 *pos* 之间的找一个油量最大的加上，然后继续向前走。这个过程用一个堆维护，如果不能走到终点则为无解。

1750: [Usaco2005 qua]Apple Catching

简单的 DP。*dp*[*i*][*j*][*k*]表示第 *i* 分钟移动了 *j* 次在第 *k* 棵树下能接到的最多苹果数量。*c*[*i*][*k*]表示第 *i* 分钟第 *k* 棵树下是否有苹果下落。*dp*[*i*][*j*][0]＝max(*dp*[*i*－1][*j*][0],*dp*[*i*－1][*j*－1][1])＋*c*[*i*][0]，*dp*[*i*][*j*][1]同理。边界条件 *dp*[0][0][0]＝0，*dp*[*i*][0][0]＝*dp*[*i*－1][0][0]＋*c*[*i*][0]。

1751: [Usaco2005 qua]Lake Counting

做一遍 floodfill 就完事了。

1752: [Usaco2005 qua]Til the Cows Come Home

真心水题，求最短路就完事了。

1755: [Usaco2005 qua]Bank Interest

这种题学 OI 第一天就应该会做了。

1756: Vijos1083 小白逛公园

线段树维护区间最大连续和，经典问题。

1758: [Wc2010]重建计划

我们考虑枚举一个根节点，那么这条路径只有两种情况：①通过根节点，②完全在根节点的某个子树中。我们发现②这种情况是可以被递归解决的，因此这道题采用分治算法。由于根节点的选择十分重要，如果选择的不好将导致下面的子树结点个数仍然非常多，因此我们每次选择树的重心作为根节点，然后分治处理。设当前的根节点为 *p*，我们考虑 *p* 的子树，我们找到与 *p* 相连的一条边 *x*，使得从 *x* 切开以后，两边的节点个数比较接近，实际操作中我找的是使得左边结点个数约等于 *size*[*p*]/2 的那条边。之后我们依次提取 *x* 左边的节点和 *x* 右边的节点，以 *x* 左边为例，我们记 *a*[*i*]表示到 *p* 的距离为 *i* 条边的最长的那条路径是多长，对于 *x* 的右边我们则记为 *b*[*i*]，那么原问题就转变成了最大化 $\frac{a[i]+b[j]}{i+j}\left(L\le i+j\le U\right)$，这个东西可能不是很好解决，但我们观察到这个表达式是一个 01 分数规划问题，所以我们想到二分答案。如果 $\frac{a[i]+b[j]}{i+j}\ge T$，则 $a[i]+b[j]-T\left(i+j\right)\ge 0$，我们考虑确定当前的 *j*，那么 *i* 的取值范围就是 $L-j\le i\le U-j$，而且随着 *j* 的减小，这个区间在整体向右移动。根据上面的算式我们有 $a[i]-Ti\ge Tj-b[j]$，所以我们就是要在 i 的取值范围内找到最大的 $a[i]-Ti$，由于决策的单调性，这个问题可以使用单调队列在 *O*(*lena*＋*lenb*)的时间内解决，其中 *lena* 和 *lenb* 分别是 *a* 和 *b* 数组的长度。这样我们就解决了这条路通过根节点的情况，然后我们把边 *x* 删掉，只要递归处理得到了两棵新的子树即可。由于重心的性质，这道题的总时间复杂度就为 *O*(*n*log*n*log*Ans*)。

1770: [Usaco2009 Nov]lights 燈

最朴素的做法显然是 $O(2^n)$，但是 n 最大为 35，会超时，采用折半搜索的方式就可以了。

1772: [Usaco2009 Nov]rescue 拯救奶牛貝希

这道题有换坐标系的神解法，请参加集训队 2006 年汪晔的论文。

1777: [Usaco2010 Hol]rocks 石头木头

答案只与深度为奇数的点有关（根节点深度为 0），把每个数对 $L+1$ 取模然后异或一下就行了。

1779: [Usaco2010 Hol]Cowwar 奶牛战争

这是一个比较难想的拆点网络流问题。设置源点 S，汇点 T。我们设 John 牛为 A，Tom 牛为 B。把草地拆成四排，建四排点。S 向第一排中所有的 A 牛连容量 1 的边，第一排中所有的 A 向第二排中对应的点连容量为 1 的边，第一排中所有的 A 向第二排中不为 B 的点连容量为 1 的边，第二排中的任意一个向第三排中对应的点连容量为 1 的边（第三排的目的是保证每个点只有一头牛），第三排中所有的不是 B 的向第四排中是 B 的连容量为 1 的边，第四排中所有的 B 向 T 连容量为 1 的边，最大流就是答案。

1786: [Ahoi2008]Pair 配对

首先可以发现最优解中，所有的−1 一定是不减的，基于这个事实我们令 $dp[i][j]$表示第 i 个改成 j 所形成的最小逆序对个数，然后就很好转移了。

1787: [Ahoi2008]Meet 紧急集合

这道题直觉告诉我，到三个点距离和最短的点，应该是两两 LCA 的其中一个。当然为什么，我也不会证 T_T。

1789: [Ahoi2008]Necklace Y 型项链

答案就是三个字符串的总长减去任意两个的公共前缀长度。

1791: [Ioi2008]Island 岛屿

容易发现每个连通分量互相独立，题目所求的就是每个连通分量选一条最长路，问这些最长路加起来是多少。

由于是 n 个点 n 条边的图，所以容易想到环套树模型。对于一个连通分量，我们首先找到环。那么最大值只有两种情况，要么通过环，要么不通过环。

对于不通过环的，也就是求若干棵树，问这些树直径的最大值的多少，这个可以用两遍 BFS 或者一遍自底向上的 DP 解决。

对于通过环的，我们不妨把环打开，比如环上节点是 1,2,3，我们打开以后变成 1,2,3,1,2,3。如果原来环上有 m 个点，则现在变成 $2m$ 个，我们设 $dp[i]$表示现在的环我们选择的路径的终点为第 i 个点能走的最长路。则有 $dp[i]=\max(down[i]+down[j]+dist[j,i])$，其中 $j-m+1<=j<=i-1$，$down[i]$表示以 i 为根的子树到 i 的最大距离是多少。我们记录 $D[i]$表示环首到第 i 个点的距离，则 $dist[j,i]=D[i]-D[j]$。带入上面的方程有 $dp[i]=\max(down[j]+D[i]-D[j])+down[i]$。用单调队列可以把转移做到均摊 $O(1)$。这样我们就在 $O(n)$的时间复杂度下解决了这道题。

1797: [Ahoi2009]Mincut 最小割

这个问题是对于每一条边有两个询问，第一个是是否可能在最小割中，第二个是是否一定在最小割中。做法就是先求一遍网络流，然后在残量网络中，也就是边权大于 0 的网络中建一个新图。然后求这个图的强连通分量。对于输入的一条边我们设为$<u,v>$，对于源点我们设为 S，对于汇点我们设为 T。对于缩点后的图，我们设第 i 个点所在的联通块为 $belong[i]$。如果 $belong[u]\neq belong[v]$，则这条边是有可能在最小割中的。如果 $belong[u]=S$ 且 $belong[v]=T$，则这条变一定在最小割中，证明也很显然，就不证了。

1800: [Ahoi2009]fly 飞行棋

暴力枚举四个点，然后判断。

1801: [Ahoi2009]chess 中国象棋

根据题意我们可以发现每行只能放置 0 个或者 1 个或者 2 个，而这行怎么样放，与前面 $i-1$ 行的具体方法并无关，只与前 $i-1$ 行有多少列放了 0 个，有多少列放了 1 个，有多少列放了 2 个有关。定义状态 $dp[i][j][k]$表示第 i 行有 j 列放了一个有 k 列放了两个的方案数，由于 j 和 k 已知，所以放了 0 个的列数就是 $m-j-k$，则 $dp[i][j][k]$由以下部分组成：

$dp[i-1][j][k]$(第 i 行放了 0 个)

$dp[i-1][j-1][k]\times(m-k-j+1)$($j>0$，第 i 行放了 1 个，放的这个格子属于之前放了 0 列的格子)

$dp[i-1][j+1][k-1]\times(j+1)$ ($j<m$ 且 $k>0$，第 i 行放了 1 个，放的这个格子属于之前放了 1 列的格子)

$dp[i-1][j-2][k]\times C(m-k-j+2,2)$($j>1$，第 i 行放了 2 个，放的这两个格子属于之前放了 0 列的格子)

$dp[i-1][j+2][k-2]\times C(j+2,2)$ ($j<m-1$ 且 $k>1$，第 i 行放了 2 个，放的这两个格子属于之前放了 1 列的格子)

$dp[i-1][j][k-1]\times(m-k-j+1)\times j$ ($j>0$ 且 $k>0$，第 i 行放了 2 个，放的这两个格子所属的列一个是之前放了 0 个的，一个是之前放了 1 个的)

答案就是 $\sum dp[n][i][j](i+j\le m)$，时间复杂度 $O(nm^2)$。

1803: Spoj1487 Query on a tree III

求出 DFS 序列以后可以发现一棵子树中所有的节点是 DFS 序中连续的一段，原问题变成一个序列$[L,R]$的第 K 小数是多少，用划分树解决。

1806: [Ioi2007]Miners 矿工配餐

很简单的 DP，$dp[i][s1][s2][t1][t2]$表示第 i 个，其中分配给 1 的前 1 个是 $s2$，前 2 个是 $s1$，$t1$ 和 $t2$ 的含义同理。那么转移就很显然了，然后对第一维用滚动数组优化，防止 MLE。

1816: [Cqoi2010]扑克牌

二分答案就行，然后乱搞。

1818: [Cqoi2010]内部白点

这道题我们不难发现如果变换的话只能变化一次，证明是这样的。假设存在点 A 是第二次变换而来，那么它必须要依赖第一次变换得来的点。我们不妨设另外的四个点是 B，C，D，E，其中有至少有一个是第一次变换而来的，但是显然第一次变换而来的点依赖的是最初的点，而这些点一定是在同一行或者同一列的，所以我们可以发现第二次变换的可以依赖最初变换的而来，那么也就是说整个变换只能有一次，并且不会出现不挺变换不中止的情况。我们把所有点的纵坐标进行离散化，那么原问题就转变了统计有多少个点，它的上下左右都有黑点，我们可以把所有的点按照 x 坐标排序，对于 x 坐标相同的点按照 y 坐标排序，然后用树状数组或者线段树来维护就可以了。时间复杂度 $O(n\log n)$。

1821: [JSOI2010]Group 部落划分 Group

二分答案以后用并查集检验就可以了。我们设当前二分的值为 S，那么如果两个点的距离小于 S 就说明他们必须在一个部落中否则不合法，最后我们看分成的部落数如果大于等于题里所设定的数量则可行，否则不可行。

1823: [JSOI2010]满汉全席

裸的 2－SAT 问题。

1827: [Usaco2010 Mar]gather 奶牛大集会

这道题比较有趣，虽然比较简单，但是不是很好想。我们首先以 1 为根节点求一次答案。如果我们知道了节点 1 的答案，则可以用节点 1 个答案去处理所有与 1 相邻的点，更新以这些点为根节点的答案。最后从所有节点中找一个最小值就可以了。具体方法就是对于相邻的节点用子树和维护。

1830: [AHOI2008]Y 型项链

跟 1789 是一个题。

1831: [AHOI2008]逆序对

容易发现－1 中填的数一定的单调不减的，这个很容易证明，考虑两个 i 和 j，如果 $i<j$ 并且 $a[i]>a[j]$那么将 $a[i]$和 $a[j]$交换位置一定可以使结果更优。然后 $dp[i][j]$表示到第 i 个数，并把它换成 j 能产生的最小逆序对数，推一遍就行了。

1834: [ZJOI2010]network 网络扩容

第一问直接求最大流就可以了。对于第二问，若一条边为 a 到 b，流量是 c，费用是 d，则建立 a 到 b，流量为 c，费用为 0 的边和 a 到 b，流量为无穷大，费用为 d 的边，求最小费用最大流就可以了。

1854：[Scoi2010]游戏

设第 i 个武器的两个属性是 p 和 q，连边 p 到 i，q 到 i，用匈牙利算法求二分图最大匹配，设 x 是第一个找不到增广路的点，则 $x-1$ 就是答案。

1856：[Scoi2010]字符串

答案就是 $C(n+m,m)-C(n+m,m-1)$。自己推推就行了。

1861：[Zjoi2006]Book 书架

直接开个 Splay 维护就行了。

1862：[Zjoi2006]GameZ 游戏排名系统

这道题和 1056 是一个题目，但是不同的是时间限制缩短，内存限制缩小。所以在 1056 中使用 C++的 STL 来维护关系已经无法在规定时间内跑出来了。我们需要把名字建成一棵 Trie 树，在查询的时候到 Trie 树上查询。出于全面考虑，假定有 25W 个不同的名字，每个名字 10 个字母，我们需要存储每个节点的后继 26 个节点（A~Z），那么这样算来会 MLE，然而考虑到名字可能很多重复，实际测试中 Trie 的节点总数开到 30W 个就足够了。

1867：[Noi1999]钉子和小球

类似于数塔问题，$dp[i][j]$表示走到 i 行第 j 列的概率。为了处理简单，我们首先设 $dp[1][1]=2^n$。那么如果当前格子可走，则用 $dp[i][j]$更新 $dp[i+1][j]$,$dp[i+1][j+1]$，否则，用 $dp[i][j]$更新在它下面第一个可以到达的格子。

1875：[SDOI2009]HH 去散步

考虑朴素的做法。$dp[i][j]$表示走了 i 条边，当前停在 j 的方案数。转移就很显然了。然后用矩阵乘法优化这个 DP。

1898：[Zjoi2004]Swamp 沼泽鳄鱼

由于每个鳄鱼的周期只能是 2 或者 3 或者 4，所以走 12 步一定可以出现一次循环。我们预处理走 12 步的情况，然后用矩阵乘法算出$12\times\left\lfloor\frac{K}{12}\right\rfloor$步的情况，对于剩下的$K-12\times\left\lfloor\frac{K}{12}\right\rfloor$步暴力求就可以了。

1911：[Apio2010]特别行动队

朴素的 DP 方程很容易写，记原序列为 $x[i]$，$s[i]=\sum_{j=1}^{i}x[j]$。$dp[i]$表示前 i 个士兵分配完，第 i 个士兵为结尾的最大战斗力，则有：$dp[i]=\max(A\times(s[i]-s[j])^2+B\times(s[i]-s[j])+C+dp[j])$，其中$(0<=j<i)$。我们考虑化简方程：

$dp[i]=\max(A\times s[i]\times s[i]+B\times s[i]+C+A\times s[j]\times s[j]-2\times A\times s[i]\times s[j]-B\times s[j]+dp[j])$。

我们设 $T[i]=A\times s[i]\times s[i]+B\times s[i]+C$，$Y[j]=A\times s[j]\times s[j]-B\times s[j]+dp[j]$，则有：

$dp[i]=\max(-2\times A\times s[i]\times s[j]+Y[j])+T[i]$，我们观察到 $s[i]$随着 i 的增加而增加，$-2\times A\times s[i]$随着 i 的增加而减小，因此可以用斜率优化做到 $O(n)$。维护一个凸包，把凸包左面不优的点删掉，然后队首元素就的当前 i 对应的决策 j，然后把点 i 加入凸包，用叉积维护一下就可以了。

1912: [Apio2010]patrol 巡逻

$K=1$ 的时候选择最长链即可。$K=2$ 的时候先选择最长链，然后把最长链上的边权设为-1，再求一次最长链。时间复杂度 $O(n)$。

1914: [Usaco2010 OPen]Triangle Counting 数三角形

我们可以求出不满足条件的三角形个数，然后用总数减去这个数得到答案。以(0,0)为中心进行极角排序，然后线性扫描一次就可以了。

1915: [Usaco2010 Open]奶牛的跳格子游戏

我们不妨倒着考虑问题，$dp[i]$表示回来的时候走到 i 的最大钱数，设一次最多可以跳 k 个格，初值是 $dp[i]=a[i]+\max(0,a[i+1])$。转移方程 $dp[i]=\max(dp[j]+a[i]+a[i+1]+sum[j-1]-sum[i+1])$，其中 $2\leqslant j-i\leqslant k$。其中 $sum[i]$表示 $a[1..i]$中所有正数的和，朴素算法时间复杂度 $O(n^2)$，可以用单调队列优化到 $O(n)$。答案是 $\max(0,dp[0],dp[1])$。0 表示不进行游戏，$dp[0]$表示正常跳回位置 0，$dp[1]$是一种特例，因为位置 1 也可以直接跳回 0。

1925: [Sdoi2010]地精部落

看网上的题解吧，有个很神的递推公式，时间复杂度 $O(n^2)$的。

1927: [Sdoi2010]星际竞速

简单费用流，建立附加源点 S，附加汇点 T。把每个点 i 拆成 i 和 i'，S 向 i 连容量为 1 费用为 0 的边，S 向 i' 连容量为 1 费用为定位时间的边，i'向 T 连容量为 1 费用为 0 的边，对于一条边$<i,j,d>$其中 $i<j$，我们由 i 向 j'连容量为 1 费用为 d 的边，求出最小费用流即可。

1930: [Shoi2003]pacman 吃豆豆

不难发现题目所求为两条不相交路径最多有多少个点，考虑用费用流解决。将每个点 i 拆成 i 和 i'，设置源点 S，汇点 T，限流点 X。S 向 i 连容量 1 费用 0 的边，i 向 X 连容量 1 费用 0 的边，X 向 T 连容量 2 费用 0 的边，

i 向 i'连容量 1 费用－1 的边。若 i 可以到达 j 则由 i'向 j 连容量 1 费用为 0 的边，最小费用的相反数就是答案。但是这题卡了 SPFA 增广，卡了内存限制，对于 SPFA 我们可以把队列改成栈即可，对于内存限制，经过测试算上反向边需要 400W 条边左右，我们把能不用 int 的变量全开成 short 就可以卡过了。

1933: [Shoi2007]Bookcase 书柜的尺寸

直接做不是很好做，我们按照高度倒序排序，这样会发现每一层的高度就只和第一个放进这层的那本书有关。用 DP 解决，我们不妨设第 1 层的高度是最高的，$dp[k][i][j]$表示到第 k 本书，第 2 层的宽度为 i，第 3 层的宽度为 j，第 2 层和第 3 层的高度和的最小值。考虑当前的 $h[i]$，$w[i]$（分别表示当前这本书的高度和宽度），则有状态转移方程：

$$dp[k][i][j]=dp[k-1][i][j]$$
$$dp[k][i][j]=\min\left(dp[k][i][j],dp[k-1][0][j]+h[k]\right)\quad i=w[k]$$
$$dp[k][i][j]=\min\left(dp[k][i][j],dp[k-1][i-w[k]][j]\right)\quad i>w[k]$$
$$dp[k][i][j]=\min\left(dp[k][i][j],dp[k-1][i][0]+h[k]\right)\quad j=w[k]$$
$$dp[k][i][j]=\min\left(dp[k][i][j],dp[k-1][i][j-w[k]]\right)\quad j>w[k]$$

，边界条件是 $dp[1][0][0]=0$，答案是 $\min_{i>0,j>0}\left((i+j+h[1])\times\max(i,j,m-i-j)\right)$，其中 m 为所有书的宽度总和，为了防止 MLE，对 DP 的第一维用滚动数组优化。

1937: [Shoi2004]Mst 最小生成树

我们考虑一条树上的边 P，一条不是树上的边 Q。不难发现如果修改的话 P 一定是减权而 Q 一定是加权，我们不妨设 P 的边权减少值为 ΔP，Q 的边权增加值为 ΔQ。当 $P\le Q$ 显然不需要修改，否则修改后的边权应该满足：$P-\Delta P\le Q-\Delta Q$，也就是说 $\Delta P-\Delta Q\ge P-Q$。我们建一个二分图，把树上的边放在左边，记为点 x_i，非树的边放在右边，记为点 y_i，那么我们就可以根据之前的信息构建出一些形如 $x_i-y_j\ge T_{i,j}$ 的不等式。建立附加源点 S，附加汇点 T，S 向 x_i 连容量为 1 费用为 0 的边，y_i 向 T 连容量为 1 费用为 0 的边。对于每个不等式由 x_i 向 y_i 连容量 1 费用为为 $T_{i,j}$ 的边。求出二分图最大权匹配就是最后的答案了。二分图最大权匹配的算法可以由费用流经过改造得到，如果不存在一个不等式 $x_i-y_j\ge T_{i,j}$，则由 x_i 向 y_i 连容量为 1 费用为 0 的边，这样只需要求最大费用最大流即可，最后的最大费用就是答案了。

1967: [Ahoi2005]CROSS 穿越磁场

把坐标离散化然后用 SPFA 求最短路，有一个点我一直 WA，没有调出来，最后骗了数据 T_T。

1977: [BeiJing2010 组队]次小生成树 Tree

一个显而易见的结论是严格次小生成树一定从最小生成树上换一条边。我们首先随便求一个最小生成树，记下哪些边在树上，哪些边不在树上。考虑一条非树边，记这条边的两个顶点分别为 a 和 b，权值为 v。我们发现如果想把这条边加上去，就需要把 a 到 b 路径上的一条边替换下来，记换下的这条边边权为 w，则此时我们求得的新生成树的边权和就比最小生成树的边权和增加了 $v-w$，显然当 w 取得最大值时，增量取得最小值，但题目说是严格次小生成树，也就是 v 不能等于 w。我们使用倍增方式进行维护，记录每个点到它 2 的 i 次方的父亲组成的路径中最大的边权多大，次大的边权多大。对于一条非树边，如果能用最大的边权换则我们就用最大的边权

换，否则如果最大的边权和这条边的边权相等，且存在次大的边权能换（次大边权存在），就用次大的边权换即可。时间复杂度 $O(n\log n+m\log n+m\log m)$。

1978：[BeiJing2010]取数游戏 game

记 $max[i]$表示俩数的最大公约数为 i 时最多能取多少个，初始情况 $max[i]=0$。对于每个 $a[i]$，用 $a[i]$的所有约数 j 去查询 $max[j]$，然后把查到的最大的 $max[j]+1$ 作为最后一个选 $a[i]$的答案，然后用 $a[i]$更新 $max[j]$，记最大数为 m，时间复杂度 $O\left(n\sqrt{m}\right)$。

1996：[Hnoi2010]chorus 合唱队

我们设原序列为 $a[i]$，排队以后的序列为 $b[i]$。我们发现 $a[1..i]$一定是 b 中的连续一段。定义状态 $dp[i][j][0]$表示得到 b 序列的$[i,j]$最后一个站在 i 的方案数，$dp[i][j][1]$是站在 j 的方案数。则有：

$dp[i][j][0]=dp[i+1][j][0]\times(b[i]<b[i+1])+dp[i+1][j][1]\times(b[i]<b[j])$；

$dp[i][j][1]=dp[i][j-1][0]\times(b[j]>b[i])+dp[i][j-1][1]\times(b[j]>b[j-1])$。

边界条件 $dp[i][i+1][0]=dp[i][i+1][1]=(b[i]<b[i+1])$。

答案是 $dp[1][n][0]+dp[1][n][1]$。

1997：[Hnoi2010]Planar

我们考虑不在哈密尔顿回路上的边，那么这条边的连法只有两种，要么直接在环内连接，要么在外面绕一圈，我们找到所有的二元组(x,y)，表示第 x 条边和第 y 条不能同时在环内或者同时在环外，然后把边看成点，连一条无向边(x,y)。我们对得到的图进行二分图染色，如果可以成功染色则可行，否则不可行。注意一个 n 个点 m 条边的图，如果 $m>3n-6$，则此图不可能是平面图，这样边数就最多只有 $3n-6$ 条，暴力建图就可以过了。

2002：[Hnoi2010]Bounce 弹飞绵羊

用动态树或者块状树做就可以了，我写的 LCT，因为块状树不会写。做法就是设一个 X(也就是第 $n+1$ 个点)如果一个点可以弹出，那么就由这个点连向 X，否则连向他弹到的那个点。对于询问操作我们只需要把 X 点 *Evert* 成根节点，然后对询问的点进行 *Access* 操作，最后把 X 点 *Splay* 到根节点，输出以 X 为根的树的节点个数减一就可以了。

2007：[Noi2010]海拔

经过观察我们可以发现，首先每个点的高度必须是整数，可以通过反证法证明。我们可以假设现在有一种方案，所有的点高度都是整数，我们通过调整一个点的高度使它变成小数，而且结果更优，我们不难发现这是一个带权中位数问题，而带权中位数总可以通过取一个与周围的点相等的值使结果最优，所以如果周围的点高度都是整数，那么这个点的高度是小数的情况下，结果不可能最优。其次，我们发现每一个点的高度只能是 0 和 1。这个证明也很容易，我们可以假想有一个点的高度特别大，与它相邻的点的高度都小于它，那么我们可以通过降低这个点的高度到与它相邻的点中高度最大的高度而使得结果变得更优。降低完成后，我们把与这个点相邻的点中与它高度相同的点看成一个点群，我们发现如果这个点群周围的点的高度都比点群的高度小，那么我们可以通过降低点群的高度使得结果变得更优。接下来由于起点的高度是 0，终点的高度是 1，如果一个点的高度降到了 1，

那么它就有可能不再会降低，因为此时有可能达到了最优高度。从上面的讨论中我们可以发现，如果所有的点高度都是 0 和 1，那么如果一个点的高度是 1，则与它相邻的点不可能高度全为 0，如果一个点的高度是 0，那么与它相邻的点不可能高度全为 1。因此在最优方案中，一定可以找到一条分界线将地图分成两部分，其中一部分全是 0，另一部分全是 1。因此我们只需要求出最小割即可，也就是最大流。但是点数可以达到 25W 个，最大流难以在 2s 内出解。我们观察原图的性质不难发现这是一个平面图，因此我们只需求出原图对偶图的最短路即可解决。

2014: [Usaco2010 Feb]Chocolate Buying

按价钱排序，水题。

2015: [Usaco2010 Feb]Chocolate Giving

水题，求一遍最短路就行了。

2016: [Usaco2010]Chocolate Eating

二分答案转变为判断性问题，注意用 64 位整数。

2018: [Usaco2009 Nov]农场技艺大赛

首先计算出 u_i 和 w_i。然后按照 u_i 从大到小，u_i 相同则 w_i 从小到大的规则进行排序。取前 n 个 w_i 加起来就是答案。千万注意计算的时候及时取模，用 64 位整数，避免溢出。

2019: [Usaco2009 Nov]找工作

我们定义一条边 i 到 j 的边权为负的输入权值加上每个点可以获得的最大利益。然后用 bellman－ford 求最长路并判断是否存在环就可以了。

2020: [Usaco2010 Jan]Buying Feed, II

这题和 2059 是一个题，唯一不同的是数据范围较小，和题目的一个细节不同。朴素的 DP 就可以过了。定义状态：$dp[i][j]$表示已经到达第 i 个商店，身上带着重量为 j 的食物所需的最小花费。则有：$dp[i][j]=\min(dp[i-1][k]+(j-k)\times cost[i]+k\times(x[i]-x[i-1]))$，其中 $k>=j-f[i]$。时间复杂度为 $O(nk^2)$。可以用单调队列优化到 $O(nk)$，具体参见 2059 即可。

2021: [Usaco2010]Cheese Towers

无穷背包问题。然后对每个物品进行枚举，将第 i 个放在最下面，更新答案。

2023: [Usaco2005 Oct]Flying Right

跟 1745 是一个题。

2028：[SHOI2009]会场预约

这道题用一棵平衡树维护就可以了。我们可以发现对于当前合法的区间一个是任意两个都没有交集的。因此我们把所有当前合法的区间扔进一棵平衡树，平衡树的关键字为区间的左端点。对于 *A* 操作，考虑等待处理区间[*x*,*y*]，我们查着平衡树中 *y* 的前驱，找到前驱对应的区间[*a*,*b*]若 *b*<*x* 则是不矛盾的，否则我们删掉区间[*a*,*b*]并重复查着前驱的操作，直到合法为止。对于 *B* 操作我们直接输出根节点的 *size* 就可以了。由于每个区间最多插入一次删除一次，时间复杂度为 $O(n\log n)$。

2038：[2009 国家集训队]小 Z 的袜子(hose)

我们把原序列分成 $n^{1/3}$ 块，记录 *sum*[*i*][*j*][*k*]表示第 *i* 块到第 *j* 块 *k* 出现的次数，*tot*[*i*][*j*]表示第 *i* 块到第 *j* 块的答案。对于一个询问[*L*,*R*]我们把它分解成[*L*,*l*)，[*l*,*r*]，(*r*,*R*]，记 *L* 属于第 *cl* 块，*R* 属于第 *cr* 块。其中中间一部分的答案我们是已知的，因为它们是一些连续块组成的，所以我们先把答案加上 *tot*[*cl*＋1][*cr*－1]。我们考虑左右两部分，以左部分为例，记录 *tmp*[*i*]表示当前询问的过程中 *i* 出现的次数，对于一个位置 *k*∈[*L*,*l*)，我们把答案加上 *tmp*[*a*[*k*]]＋*sum*[*cl*＋1][*cr*－1][*a*[*k*]]，这是当前这个数对分子的贡献，然后把 *a*[*k*]加 1。当询问结束的时候，我们只需要把 *a*[*k*]置零，即可将 *tmp* 数组置零，这样我们就得到了分子的值，而分母就很简单了，分母就是 *C*(*R*－*L*＋1,2)，然后约分一下输出就行了。时间复杂度 $O(n^{5/3})$。

2048：[2009 国家集训队]书堆

物理题，$Ans = m\times\sum_{i=1}^{n}\frac{1}{2i}$，当 *n* 比较小的时候暴力算，否则考虑调和级数。

2049：[Sdoi2008]Cave 洞穴勘测

裸的动态树问题，用 Link_Cut_Tree 就行了，刚刚学会……

2058：[Usaco2010 Nov]Cow Photographs

我们记录 *i* 出现的位置为 *loc*[*i*]。首先我们考虑这样一个过程，将原数列变为 1,2,……*n*。这需要的操作部数为原序列的逆序对的个数，这一步很好想。接下来我们考虑如何把 1,2,……*n* 变为 2,3,……*n*,1。这一步的操作次数为 *n*－*loc*[1]－(*loc*[1]－1)，剩下的也同理了，最后取一个最小的就行了。

2059：[Usaco2010 Nov]Feed 购买饲料

显然这是一道动态规划。状态：*dp*[*i*][*j*]表示到达第 *i* 个商店，身上带重量为 *j* 的食物所需的最小花费。转移：*dp*[*i*][*j*]＝min(*dp*[*i*－1][*k*]＋(*j*－*k*)×*cost*[*i*]＋*k*×*k*×(*x*[*i*]－*x*[*i*－1]))，其中 *k*>＝*j*－*f*[*i*]。朴素的动规会超时。我们对方程进行整理有：*dp*[*i*][*j*]＝min(*dp*[*i*－1][*k*]－*k*×*cost*[*i*]＋*k*×*k*×(*x*[*i*]－*x*[*i*－1]))＋*j*×*cost*[*i*]。其中 min 的表达式中 *k* 的取值范围随着 *j* 的增加单调不减，维护一个单调队列即可。时间复杂度为 $O(nk)$。

2060: [Usaco2010 Nov]Visiting Cows 拜访奶牛

这道题的意思是选出树上尽可能多的点，使得任意两点间没有边相连。做法就是 TreeDP，首先我们进行一次 BFS，求出一个深度序列，然后按照这个序列的倒序求解。状态 $dp[i][0]$表示以 i 为根的树不选第 i 号节点的能得到的最大结果，$dp[i][1]$表示以 i 为根的树选第 i 号节点能得到的最大结果。则有：$dp[i][0]=\Sigma\max(dp[son[i]][0], dp[son[i]][1])$，$dp[i][1]=\Sigma dp[son[i]][0]+1$。答案则是 $\max(dp[1][0],dp[1][1])$。

2079: [Poi2010]Guilds

单独考虑每个连通分量，易知当连通分量只有一个点时一定无解。否则我们随便在连通分量内求一棵生成树，然后把奇数层的答案设为 S，偶数层的答案设为 K，这一定是满足题目条件的。

2080: [Poi2010]Railway

我们设从栈底到栈顶的数依次为 $a[1..n]$，就是按照题目给定的输入顺序编号。接下来我们考虑什么情况下 i 和 j 不能进入同一个栈，我们发现当 $i<j$ 时，如果 $a[i]<a[j]$，并且存在一个 k 使得 $j<k$ 且 $a[k]<a[i]$，也就是说 $i<j<k$ 且 $a[k]<a[i]<a[j]$，那么显然第 i 个数和第 j 个数不能进入同一个栈。证明也非常的简单，假设可以进同一个栈的话，在 $a[i]$进栈时，由于存在比它小的 $a[k]$，所以 $a[i]$只有等 $a[k]$出栈了 $a[i]$才能出栈，但是当 $a[j]$入栈以后 $a[i]$就被压在了下面如果 $a[i]$想出栈则 $a[j]$必须先出栈，但是如果 $a[j]$先出栈就会导致大的数比小的数先出栈，这显然是不合法的。因此我们就得到了一个 $O(n^2)$的做法，我们建一个无向图，把所有不合法的(i,j)之间连一条边，这样就变成了判断该图是否是一个二分图。

上述算法的瓶颈就在于图的边数可能非常大，而我们发现，如果要把一个图进行二染色，只需要求出这个图的一棵生成树，然后按照奇数层染一个颜色，偶数层一个颜色就可以了。因此我们不妨考虑构建一棵生成树，然后把图进行二染色再判断这个方案是否可行。

我们设 $b[i]=\min(a[k])$，$i<k<=n$。则上述条件中 $i<j<k$ 且 $a[k]<a[i]<[j]$就变成了 $i<j$ 且 $b[j]<a[i]<a[j]$。我们可以发现对于一个 $a[j]$，有用的 $a[i]$范围是在$[b[j]+1,a[j]-1]$之间的，而且从 $b[i]$的定义我们可以发现随着 i 的增加，$b[i]$也是单调不减的，因此区间$[b[j]+1,a[j]-1]$的左端点是随着 j 的增加是单调不减的。

我们考虑从 1 到 n 枚举 j，对于当前的 j 做以下三件事情：

①. 把所有小于等于 $b[j]$的数删除。

②. 把所有小于等于 $a[j]-1$ 组成的数与 j 连边。

③. 把数 $a[j]$插入。

上面的算法中的第②步，边数仍然可以达到 $O(n^2)$。我们继续考虑如何优化。

我们不妨对每个元素新建一个集合，这样②就变成了把所有存在小于等于 $a[j]-1$ 的集合与 j 连一条边，然后合并这些集合。这样我们只需要对每个元素开一个堆，合并的时候使用启发式合并即可。时间复杂度 $O(n\log^2 n)$。

2081: [Poi2010]Beads

首先与处理原串的 HASH 值以及把原串反转后的反串的 HASH 值。然后我们枚举每个 K，对于给定的 K，暴力算出 n/K 段的正反 HASH 值，并把它们存到一个数组里，然后排序这个数组，去重并计算答案。时间复杂度 $O\left(\sum_{i=1}^{n} n/i\log(n/i)\right)\approx O(n\log n)\sim O(n\log^2 n)$。

2082: [Poi2010]Divine divisor

我们可以发现，如果我们能把每个 $a[i]$分解质因数，那么问题就可以很容易地解决，但是 $a[i]$很大，暴力分解显然不行。

我们考虑筛出 1 到 10^6 的所有素数。然后先对每个 $a[i]$用这些素数进行试除，记录每个素数出现的个数，以及每个 $a[i]$除完后的结果。我们发现尽管试除后的 $a[i]$可能不为 1，但是 $a[i]$一定可以写成 p 或者 p^2 或者 pq（其中 p 和 q 均为大质数）的形式，因为 $a[i]$最多只能是两个素数的乘积，如果是三个或者以上的话 $a[i]$必然大于 10^{18}，这与数据范围矛盾。对于 $a[i]=p$，我们可以用 *miller－rabbin* 处理掉，对于 $a[i]=p^2$，我们可以把 $a[i]$开根号然后判断掉。因此我们只需要考虑 $a[i]=pq$ 的情况。我们把 $a[i]=p$ 和 $a[i]=p^2$ 中的 p 全部取出并放入数组 A，然后用 $a[i]=pq$ 去试除取出的 p，如果能除则把 $a[i]/p$ 也加入到 A，直到对于任意的 $a[i]$都不能整除 p，可以发现这个过程最多重复两次。处理完成以后我们就保证了对于剩下的任意一个 $a[i]=pq$ 涉及的两个素数 p 和 q 与我们之前素数的次幂毫不相关。然后我们枚举 $a[i]$和 $a[j]$，如果 $a[i]$和 $a[j]$的最大公约数不是 1 且 $a[i]$不等于 $a[j]$，则我们就成功分解了 $a[i]$和 $a[j]$。处理完这些以后，剩余的 $a[i]=pq$，对于任意两个 $a[i]$和 $a[j]$，要么是 $a[i]=a[j]$，要么是 $a[i]$和 $a[j]$互质。到这里我们把之前的分解信息处理完成，统计每个素数的出现次数，设出现次数最多的素数出现了 a 次，一共有 b 个这样不同的素数。则第一问的答案就是 a，第二问的答案就是 2^b-1，第二问需要写个高精度来输出答案。

2083: [Poi2010]Intelligence test

由于每个数的不超过 10^6 的，我们开个 ***vector*** 记录所有等于 i 的数出现的每个位置。对于每个询问，只需要按照给出的数依次找到比上一个数位置大的那个最小的位置在哪里即可，找不到就是不合法了。记原序列有 n 个数，询问的数一共有 m 个，时间复杂度 $O(m\log n)$。

2084: [Poi2010]Antisymmetry

可以发现串的长度一定是偶数，所以原问题变成询问有多少长度为偶数的字符串，使得从这个字符串的中间切开，得到两个小字符串，把右面的取反之后，这个串是回文串。这个问题可以用后缀数组或者 HASH 很容易做到 $O(n\log n)$解决。我们考虑更快的做法：在相邻两个字符之间添加一个不为 0 也不为 1 的特殊字符，这样我们要求的长度为偶数的回文串，实际上就是以这个特殊字符为对称中心的回文串个数，接下来使用 *manacher* 在 $O(n)$ 的时间内就可以解决了。

2085: [Poi2010]Hamsters

我们首先根据题目给出的单词处理出数组 $a[i][j]$表示在第 i 个单词后面最少加多少个字母能组成单词 j，这个用 HASH 暴力处理一下就可以了，特判当 $i=j$ 时，$a[i][j]$为第 i 个单词的长度减一。然后我们可以写出朴素的 DP 方程，$dp[i][j]$表示已经出现了 i 个单词，最后一个单词是第 j 个单词所需要的最小长度，则 $dp[i][j]=\min(dp[i-1][k]+a[k][j])$。朴素的做法显然超时，我们考虑如何优化这个方程。

我们发现如果是 $dp[i][j]=\Sigma(dp[i-1][k]\times a[k][j])$，那么这就可以用矩阵乘法进行加速。类似的，我们重新定义矩阵乘法中 $c=\Sigma(a_i\times b_i)$的运算规则，我们令 $c=\min(a_i+b_i)$，然后用矩阵乘法加速这个方程就可以了。记字符串总长度为 p，时间复杂度为 $O(np+n^3\log m)$。

2086: [Poi2010]Blocks

我们考虑一个询问 k，我们把每个 $a[i]$都减去 k。设 $a[i]=a[i]-k$，则题目变成每次可以选择一个正的 $a[i]$，然后把 $a[i]$减去 1，把 $a[i-1]$或者 $a[i+1]$加上 1，问最后最长的一段非负的数有多长。记 $s[i]=\sum_{j=1}^{i} a[j]$，$sum(i,j)=s[j]-s[i-1]$。首先我们给出结论，如果 $sum(i,j)\geqslant 0$，则区间$[i,j]$是满足题目条件的。证明也非常容易，我们在$[i,j]$中选择一个正数和一个负数，比如位置分别为 x 和 y，不妨设 $x<y$，则我们把 $a[x]$减 1，$a[x+1]$加 1，然后把 $a[x+1]$减 1，把 $a[x+2]$加 1，这样不断进行下去，一定可以把从 $a[x]$那里拿来的 1 转移到 $a[y]$上，又因为 $sum(i,j)$是非负的，因此一定经过若干次操作使得每个数都是非负的。现在我们的任务就变成了找最长的区间$[i,j]$满足 $sum(i,j)\geqslant 0$。其中一种可行的做法是递增枚举 j，然后问题变成找$[0,j-1]$内最靠前的一个 i 使得 $sum[j]\geqslant sum[i]$，这个可以通过维护一个单调栈，然后在栈中二分查找，时间复杂度为 $O(mn\log n)$，优化一下常数也是能过的。

我们考虑时间复杂度更低的做法，我们以 $sum[j]$为例，如果 $j_1<j_2$ 且 $sum[j_1]\leqslant sum[j_2]$，那么 j_1 一定是没有用的，j_2 一定比 j_1 更优。类似的，i_1 和 i_2 也是同理。因此我们可以先把 $sum[j]$放进一个单调栈，栈中的元素下标递减，sum 递增。然后类似地对 i 也维护一个单调栈，栈中的元素满足下标递增，sum 递减，然后就很显然的可以通过 $O(n)$的时间归并扫描两个单调栈得出答案。时间复杂度 $O(nm)$。

2087: [Poi2010]Sheep

我们一条对角线(i,j)连接以后，如果分出的两半存在奇数个点，那么这条对角线无论如何都不能出现在答案的方案中，原因是奇数再分割会变成一个奇数和一个偶数，这样就还会有一个奇数，这个奇数无论如何都无法被消掉，因此我们考虑预处理每条对角线(i,j)连接以后分成的两半点是否都是偶数。我们考虑枚举点 i 的位置，将所有的羊按照以 i 为中心进行极角排序，记共有 m 头羊，然后就可以在 $O(n+m)$的时间统计出每个点是否合法。注意一个特殊的地方，有一只羊在这条对角线上，那么这条对角线也是不合法的。极角排序的时间复杂度为 $O(m\log m)$，因此每个位置 i 的统计时间为 $O(m\log m)$。得到这个以后我们定义 $dp[i][j]$表示顺时针方向上的第 i 个点到第 j 个点的方案数，转移方程是 $dp[i][j]=\Sigma dp[i][k]\times dp[k][j]$，其中 $i<k<j$，当 i 和 j 为不合法的对角线时有 $dp[i][j]=0$，边界条件是 $dp[i][i+1]=0$，最终的答案是 $dp[1][n]$。时间复杂度 $O(n^3+nm\log m)$。

2088: [Poi2010]Teleportation

我们设按照最优方案加边得到的图为 G。易知 G 中到点 1 的距离只能为 0,1,2,3,4,5，到点 2 的距离也是如此。因此我们考虑把 G 中的点进行分组，不考虑点 1 和点 2，设到 1 距离为 1 的点组成集合 A，到 1 距离为 2 的点组成集合 B，到 2 距离为 2 的点组成集合 C，到 2 距离为 1 的点组成集合 D，剩下的 $n-2-A-B-C-D$ 组成集合 E。我们发现：

①.相邻两个集合之间加边不影响最短路，也就是说点 1 向 A 连边，A 向 B 连边，B 向 C 连边，C 向 D 连边，D 向 2 连边。

②.同一集合内的点两两连边不影响最短路，也就是 $A\times(A-1)/2$，$B\times(B-1)/2$ 等等。

上两部分对答案的贡献为 $A+A\times B+B\times C+C\times D+D+A\times(A-1)/2+B\times(B-1)/2+C\times(C-1)/2+D\times(D-1)/2$。

接下来我们考虑集合 E 中的点，我们发现集合 E 中的点一定是通过连边进入 A，B，C，D 四个集合中的一个：

如果进入 A 集合，每个 E 中的点可以向 A 和 1 和 B 连边；

如果进入 B 集合，每个 E 中的点可以向 B 和 A 和 C 连边；

如果进入 C 集合，每个 E 中的点可以向 C 和 B 和 D 连边；

如果进入 D 集合，每个 E 中的点可以向 D 和 C 和 2 连边；

易知每个 E 中的点都选上面四种情况的最大值就是最优解，记 $x=\max(1+A+B,A+B+C,B+C+D,C+D+1)$。则这一步对答案的贡献为 Ex。

因此总答案就位上面三部分的和减去 m，其中 m 为数据给定的初始边数。时间复杂度 $O(n+m)$。

2089: [Poi2010]Monotonicity 1 & 2

朴素的动规方程很好写，设 $dp[i]$表示序列的最后一个数为第 i 个数时的最大长度，$dp[i]$可以由三部分转移而来，令 $0<=j<i$，则有：

①. $dp[i]=\max(dp[j])+1$，长度为 $dp[j]$的序列后面为一个小于号，$a[j]<a[i]$；

②. $dp[i]=\max(dp[j])+1$，长度为 $dp[j]$的序列后面为一个等于号，$a[j]=a[i]$；

③. $dp[i]=\max(dp[j])+1$，长度为 $dp[j]$的序列后面为一个大于号，$a[j]>a[i]$。

这样我们就有了 $O(n^2)$时间复杂度的做法。由于每个数都是不超过 10^6 的，因此我们可以考虑从这来优化上面的方程。对于②情况，我们开个数组 $maxlen[i]$表示数 i 的最大长度为多少，这样我们每次只需要调用 $maxlen[a[i]]$ 即可完成②的转移，然后更新 $maxlen[a[i]]$，这样②部分的每次转移代价就变成了 $O(1)$。由于①和③情况类似，所以我们只考虑①。开一棵线段树，节点 i 表示值为 i 的数对应的长度最长为多少，对于一个 $a[i]$，我们只需要询问线段树中 0 到 $a[i]-1$ 的最大值即可，然后把这个最大值加 1 即可完成 $dp[i]$的转移，然后在线段树中更新节点 $a[i]$的值，记出现的最大数为 m，这样①和③的转移就变成了每次 $O(\log m)$。在动规的时候记下每个数是从前面的哪个数转移而来，即可完成方案的输出。时间复杂度 $O(n\log m)$。

2091: [Poi2010]The Minima Game

我们发现最优策略一定是先取最大的，然后按照从大到小的顺序取。我们先把每个数从小到大排序，状态 $dp[i]$表示前 i 个数，先手能比后手最多多多少。则转移方程为 $dp[i]=\max(dp[i-1],a[i]-dp[i-1])$。其中 $dp[i-1]$表示在取 $i-1$ 的时候顺便把第 i 个也带上，$a[i]-dp[i-1]$表示把状态 $dp[i-1]$交给后手，然后自己取 $a[i]$，这样差就变成了 $a[i]-dp[i-1]$。由于排序的存在，时间复杂度 $O(n\log n)$。

2093: [Poi2010]Frog

我们考虑先预处理 $next[i]$表示第 i 个位置跳 k 个会跳到哪。我们考虑维护 $L[i]$和 $R[i]$，表示对与 i 来讲离它最近的 $k-1$ 个位置组成区间$[L[i],R[i]]$，满足 $R[i]-L[i]+1=k$，虽然这个区间是 k 个位置，但是因为必然存在第 i 个位置本身，所以实际上有用的是 $k-1$ 个。第 k 个位置就在 $L[i]-1$ 和 $R[i]+1$ 中选一个较优的，注意当 $L[i]=1$ 或者 $R[i]=n$ 时需要进行特判，由于 $L[i]$随着 i 的增加是单调不减的，因此可以在 $O(n)$是时间内处理出 $next[i]$。

处理完这个以后我们就得到个每个格子跳 1 次到达哪里，然后只需要用倍增的方式即可求出每个格子跳 m 次走到哪里，倍增要使用滚动数组，否则空间不够。时间复杂度 $O(n\log m)$。

其实有更快的做法，我们注意到如果把 i 向 $next[i]$连一条边，则整个图构成了若干个连通分量，每个连通分量都是一棵树上面长一个环，树是内向树。因此我们只需要判断每个位置跳 m 次是否跳到环上，如果没跳到就停下即可，否则一定是在环上饶若干圈，然后停在环上的某个位置。如果我们在遍历图的时候记下每个点往跳 m 个点到达树的哪里（或者已经到了环上），然后记录环长和每个环每个节点的标号，时间复杂度就可以做到 $O(n)$，但是 $O(n\log m)$的做法就以及可以通过所有测试数据了。

2095: [Poi2010]Bridges

首先我们二分答案，然后考虑检验当前的 mid 是否可行。对于题目给定的无向边，如果一条边的正反两个权值都小于等于 mid，则这条边还是无向边，如果其中一个大于 mid，则这条边变为有向边，如果两个权值都大于 mid 则显然 mid 这个答案不可行，因为不可能经过这条边。接下来就变成了判断一个混合图是否存在欧拉回路，如果有的话还要输出一种方案。这个有很经典的网络流做法：

首先我们把每条无向边任意规定一个方向，然后统计每个点的入度和出度，如果一个点的出入度之差不能被 2 整除则显然不合法，因为不管怎么调整都是出入度一个加 1 一个减 1，因此出入度之差的奇偶性始终是保持不变的，而我们知道一个有向图存在欧拉回路当且仅当每个点的出入度差都是 0，因此如果出入度差为奇数则显然无论怎么调节无向边的方向都不存在一组可行解。接下来记点 i 的出度减去入度为 $deg[i]$，S 向每个 $deg[i]$为正的点连容量为 $deg[i]/2$ 的边，每个 $deg[i]$为负的点向 T 连容量为 $-deg[i]/2$ 的边，忽略已经定向的有向边，对于之前我们随便选定方向的每条无向边，如果规定方向是 i 到 j，则由 i 到 j 连容量为 1 的边。记所有 $deg[i]$值为正的点的所有 $deg[i]$之和为 k，如果最大流不等于 $k/2$ 则无解，否则有解。如果我们之前添加的边(i,j)的容量由 1 变成了 0，则说明边(i,j)最初的定向是错误的，需要反向，确定了所有边的方向以后只需要进行一次 *dfs* 即可找到欧拉回路。

2096: [Poi2010]Pilots

我们考虑顺序枚举序列的右端点 i，设在位置 i 处，取得最长合法序列的左端点为 $L[i]$，易证随着 i 的增加 $L[i]$单调不减，于是开两个单调队列维护最大值和最小值即可，注意一下内存限制。时间复杂度 $O(n)$。

2097: [Usaco2010 Dec]Exercise 奶牛健美操

不难想到要二分答案，接下来我们考虑如何检验一个答案是否可行。我们 dfs 整棵树，定义函数 $getdp(p,d)$，表示返回以 p 为根的子树，距离 p 不超过 d 的距离的最大距离。当 p 为叶子节点时，返回 0。否则我们取得 p 的所有子树的返回值，并把每个返回值都加 1，这样就求出了到 p 的最大距离。开一个 vector 记录这些距离，我们把距离大于 d 的都删掉并答案加 1，然后我们维护最大值和次大值，如果这两个数的和大于 d 则删掉最大值并答案加 1 然后重复该过程，否则退出。最后如果答案不超过题目的限制则可行，否则不可行。

2099: [Usaco2010 Dec]Letter 恐吓信

我们把两个字符串连接在一起，中间用一个不同于 A~Z 的字符隔开，跑一遍后缀数组。由于文章有无数个，我们只需要把每次在原串中能找到的最长的一段加入即可。直到当前的指针跑出等待匹配的字符串。

2100: [Usaco2010 Dec]Apple Delivery

也是水题，求一遍最短路就行了。SPFA 注意需要加 SLF 优化，否则最后一个点跑不出来，是故意卡 SPFA 的数据。

2101: [Usaco2010 Dec]Treasure Chest

和 USACO Training 中一道一样的 DP。$dp[i][j]$表示从$[i,i+j]$可以取得的最大值。边界条件是 $dp[i][0]=a[i]$。我们记 $s[i]$表示 $a[1..i]$的和。则有 $dp[i][j]=s[i+j]-s[i-1]-\min(dp[i][j-1],dp[i+1][j-1])$。对第二维用滚动数组优化，防止 MLE。

2102: [Usaco2010 Dec]The Trough Game

2^n 枚举就行了。不用剪枝也能过。

2111：[ZJOI2010]Perm 排列计数

由 $a[i]>a[i/2]$我们可以观察到如果把 $a[1]$到 $a[n]$，按照二叉树来排的话那么构成的一定是一个小根堆。我们记 $s[i]$表示以 i 为根的子树的节点个数，那么答案就是 $\frac{n!}{\prod_{i=1}^{n} s[i]}$。

2120：数颜色

首先离散化，把修改和原数列离散成不超过 $n+m$ 的数。然后分成 $n^{1/3}$ 块，记 $tot[i][j][k]$表示第 i 块到第 j 块 k 出现了多少次，记 $sum[i][j]$表示第 i 块到第 j 块的答案。时间复杂度 $O(n^{5/3}+m\times n^{2/3})$。

2124：等差子序列

我们可以发现如果长度为 d 的合法则长度为 3 的也一定合法，因此我们只要找到一个长度为 3 的即可认为存在题目所求的等差数列。我们顺序扫描整个序列，记第 i 个数为 $a[i]$，则如果 $a[i]-d$ 扫描过且 $a[i]+d$ 没扫描过则我们就找到了以 $a[i]$为中间数的等差数列，我们开一个长度为 n 的二进制数组，记第 i 位为 1 表示 i 这个数扫描过。则我们发现如果 $a[i]-d$ 和 $a[i]+d$ 位不同时为 1 则 $a[i]$合法。但是这个判断比较麻烦，我们不妨考虑什么时候不合法，可以发现只有对于任意的 $a[i]-d$ 和 $a[i]+d$ 都有 $a[i]-d$ 和 $a[i]+d$ 同时为 0 或者同时为 1 的时候 $a[i]$才不合法，否则 $a[i]$合法。具体实现就是我们可以把长度为 n 的二进制数建成一个 HASH，然后维护正序和逆序两个 HASH 值，查询时候就变成了判两个子串是否相等。时间复杂度 $O(Tn\log n)$。

2142：礼物

易知答案为 $C_n^{w_1}\times C_{n-w_1}^{w_2}\times C_{n-w_1-w_2}^{w_3}\times\ldots\times C_{n-w_1-w_2-\ldots-w_m}^{w_m}$，当 $n<\sum w_i$ 时无解，我们只考虑有解的情况。对上面的组合数进行化简得到答案为 $\frac{n!}{w_1!w_2!\ldots w_m!\left(n-\sum w_i\right)!}$，由于取模的数 P 不是素数，所以我们考虑把取模的数分解，设分解后的结果为 $P=\prod p_i^{c_i}$，我们逐个考虑计算 $\frac{n!}{w_1!w_2!\ldots w_m!\left(n-\sum w_i\right)!}$ 对 $p_i^{c_i}$ 的模，这个可以通过把每个数分为含有 p_i 这个因子和不含有 p_i 这个因子两个部分来考虑。不妨记 $T=\frac{n!}{w_1!w_2!\ldots w_m!\left(n-\sum w_i\right)!}$，则我们得到了 r（其中 r 为 P 的质因数个数）个形如 T 对 x 的模为 y 的同余方程，接下来使用中国剩余定理暴力合并即可。

2145：悄悄话

这题只能乱搞了，贴一个有 2000 个单词的单词表，再贴一个英文字母的出现频率表，然后自己设置一些估价函数就可以了。

2172：Mario 填格子

设左上角为 n，右下角为 m，如果右下角的数不是左上角的倍数显然不可行，然后一个显然的观察就是我们可以把方格的每个数都除以 n，我们设此时的 m 是除以 n 以后的 m。我们把 m 分解质因数，设共有 k 个不同的质因子，第 i 个质因子有 a_i 个，我们考虑按如下方式构造答案：

当 $k \geq 4$ 时，不妨设 $m = abcd$ ：

1	a	ad
b	ab	abd
bc	abc	$abcd$

当 $k = 3$ 时，如果每一个质因子的次数都为 1 则无解，否则有解，不妨设 $m = a^2bc$ ：

1	a	a^2
b	ab	a^2b
bc	abc	a^2bc

当 $k = 2$ 时，有两种情况有解，第一种是每个质因子次数都大于等于 2，不妨设 $m = a^2b^2$ ：

1	b	b^2
a	ab	ab^2
a^2	a^2b	a^2b^2

第二种情况是其中一个质因子次数大于等于 5，不妨设 $m = a^5b$ ：

1	a^2	a^3
a	a^4	a^5
a^3b	a^4b	a^5b

当 $k = 1$ 时，唯一出现质因子次数大于等于 8 则有解，否则无解，不妨设 $m = a^8$ ：

1	a	a^2
a^3	a^4	a^5
a^6	a^7	a^8

接下来我们考虑如何把 m 分解质因数，首先我们用 1 到 10^6 的数进行试除，如果没除干净，剩下的 m 一定是 p 或者 pq 或者 p^2 这三种形式之 1（其中 p 和 q 均为大于 10^6 的质数）。对于第一种可以用 *miller－rabin* 判断掉，第三种可以把 m 开根号再平方看和 m 相不相等，如果既不是第一种也不是第三种，那么剩下的就是第二种了。

2173：整数的 lqp 拆分

找规律就行了，$f[i]$表示拆 i 的方案数，显然 $f[1]=1$，$f[2]=2$，$f[i]=2\times f[i-1]+f[i-2]$。

2179：FFT 快速傅立叶

裸的 FFT，会模板就能做。

2194：快速傅立叶之二

由题意得：$c_k=\sum a_i b_{i-k}$，设数组下标范围为[0,n)，我们考虑把 b 数组反过来，则有 $c_k=\sum a_i b_{n-1-i+k}$，我们发现 a 和 b 下标是与 k 成正相关的一次函数，我们重新定义 c 的值，有 $c_{n-1+k}=\sum a_i b_{n-1-i+k}$，不难发现这个值是一个卷积形式，用 FFT 加速即可，时间复杂度 $O(n\log n)$。

2196：[Usaco2011 Mar]Brownie Slicing

首先二分答案，然后贪心去切行，对于第[l,r]行再贪心切列即可。

2197：[Usaco2011 Mar]Tree Decoration

自底向上进行 TreeDP，记 $dp[i]$表示以 i 为根的子树满足条件的最小装饰代价，$sum[i]$表示以 i 为根的子树满足条件的装饰总个数，$minv[i]$表示以 i 为根的子树的所有节点装饰单价的最小值，记 $\boldsymbol{v}$ 为 p 的儿子集合，则有 $dp[p]=\Sigma(dp[\boldsymbol{v}])$，$sum[p]=\Sigma(sum[\boldsymbol{v}])$，$minv[p]=\min(minv[\boldsymbol{v}],cost[p])$。若 $sum[p]<need[p]$则把 $dp[p]$加上$(need[p]-sum[p])\times minv[p]$，然后把 $sum[p]$赋值为 $need[p]$。

2199：[Usaco2011 Jan]奶牛议会

首先用 2－SAT 判是否有解。对于有解的部分，我们枚举每一个点是使用 Y 还是使用 N，如果推出矛盾那就说明是不合法的。

2200：[Usaco2011 Jan]道路和航线

这道题的正解应该是先用 Tarjan 求出联通块，对于块内的点进行 Dijkstra＋Heap，对于块外的点进行 TreeDP。但是我用 SPFA 就给骗过去了= =。说一下技巧，首先 SLF 优化是必须要加的，其次在进行 SPFA 的过程中，比如 2 号点，它连向很多条边，我们按照边权从小到大的顺序进行扫描就可以过了，这个顺序是可以预处理的。这两个优化缺一个貌似都过不了。

2208：[Jsoi2010]连通数

直接做不好做，先缩环，然后记录每个强连通分量中点个数，然后按拓扑序倒序统计一下。对于两个连通分量 i 和 j，如果 i 可以到 j，那么答案增加 $size[i]\times size[j]$。

2209: [Jsoi2011]括号序列

把左括号看成是 1，右括号看成是－1，然后建一个 Splay 使劲搞吧，差点写吐血了。

2212: [Poi2011]Tree Rotations

我们设 $f[i]$表示以 i 为根的子树逆序对的最小值，则有 $f[i]=f[lch]+f[rch]+merge(lch,rch)$。我们观察到 $merge(lch,rch)$只有两种情况，第一种是不旋转，第二种是旋转，记 lch 的大小为 $sizelch$，rch 的大小为 $sizerch$，如果不旋转的答案为 x，则旋转的答案就位 $sizelch\times sizerch-x$，如果旋转的答案为 x，则不旋转的也同理。因此我们只需要求出 x 的值即可，这个可以通过平衡树启发式合并的方式，把 $size$ 较小的一棵树合并到 $size$ 较大的一棵，合并的同时即可算出 x 的值。时间复杂度 $O(n\log^2 n)$。至于另一道 n 达到 10^6 的题，只需用相同的方法，然后再优化一下常数即可得到满分。

2213: [Poi2011]Difference

我们把每个字母出现的位置单独拿出来，用一个 *vector* 存，然后去枚举出现次数最多的字母和出现次数最少的字母，把这两种字母按照输入的顺序取出，把出现次数最多的字母看成是 1，出现次数最少的字母看成是－1，这样问题就变成了求一个序列的最大连续和，但题目要求出现次数最少的字母不能出现 0 次，因此这个最大连续和组成的序列必须至少包含一个－1。我们考虑当前扫描到第 i 个位置，则第 i 个位置能提供的最大连续和为 $sum[i]-\min(sum[j])$，其中 $j+1$ 到 i 之间至少有一个－1，我们发现随着 i 的增加，j 的范围也在增加，而且 j 的可行范围一定是$[0,j]$因此我们可以考虑维护 j 的位置，这样就维护了 $\min(sum[j])$。时间复杂度 $O(26n)$。

2214: [Poi2011]Shift

我们考虑合并两个操作，合并后的功能为可以将一个数左移两位，这个很好证明，比如想把位置 x 的数左移两位只需要先调用（A）$n-x+3$ 次，然后调用一次（B），然后再调用（A）$x-3$ 次即可。我们考虑按照顺序构造答案，假设 1 到 $i-1$ 已经回到了 1 到 $i-1$ 的位置上，现在考虑恢复第 i 个数，设第 i 个数目前所在的位置为 j，我们发现如果 i 与 j 的奇偶性相同，那么就只需要把 j 左移两位直到到达 i 为止，如果 j 和 i 的奇偶性不同，那么我们就把 $j+1$ 位置上的数左移两位，这样 i 和 j 的奇偶性就又相同了，重复上面的步骤即可。对于当 $i=n-1$ 且 i 没有恢复时，需要特判，因为此时已经不存在 $j+1$ 这个位置了，这时序列的后三位一定是 $n-2$，n，$n-1$，如果 n 为一个奇数的话那么是无解的，否则我们把 $n-1$ 这个数，也就是第 n 个位置上的数，左移 $n/2$ 次就可以了。时间复杂度 $O(n^2)$。

2215: [Poi2011]Conspiracy

我们发现对于任意一个合法方案，记集合 A 表示里面的点两两有边，集合 B 表示里面的点两两无边，记输入文件中点 i 的度数为 $d[i]$，则对于任意的 $i\in A$，$j\in B$，必有 $d[i]>=d[j]$，否则的话一定不是一个合法的方案。我们记 A 中的点度数总和为 SA，B 中的点度数总和为 SB，A 中的点个个数为 K，则必有 $SA-SB=k\times(k-1)$。因此我们可以考虑把所有点按照度数从小到大进行排序，并记录点度数的前缀和。则对于$[i+1,n]$和$[1,i]$这两部分点能否构成一个合法的方案就可以在 $O(1)$的时间内进行判断，我们设度为 k 个点有 n 个，需要分到集合 A 中 m 个，

由于这 k 个点是等价的，所以答案加上 $C(n,m)$即可。

2216: [Poi2011]Lightning Conductor

我们反过来考虑问题，则 $h[i]+p\geq h[j]+\sqrt{|i-j|}$。我们考虑一个位置 j 对位置 i 的贡献。我们不妨假设 $i<j$，对于 $i>j$ 的情况可以类似处理。我们设使 i 取得最大 p 的位置为 $b[j]$，则我们可以发现如果新增了一个 j，j 能更新的 $b[i]$的范围一定是某个点一直到 j，这一定是一个连续的区间。因此我们可以考虑用栈来维护整个过程，每个栈元素记录$[L,R]$和 d，表示第 L 到第 R 个数的 p 取得最大值的位置是在 d 位置。每新增一个 j，我们可以二分能影响的最大范围，找到对应的块，然后在块内继续二分，找到具体位置，然后修改栈元素。时间复杂度 $O(n\log n)$。

2217: [Poi2011]Lollipop

可以观察到如果 x 合法，则 $x-2$ 一定合法，证明如下：
我们不妨设组成 x 的序列为 A.....B。如果 $A=B=2$，则取 A.....或.....B 即可构造出 $x-2$，如果 $A=1$ 则取 A.....，如果 $B=1$，则取.....B，如果 $A=B=1$，则取.....。因此我们只要记录总和 S，则与 S 同奇偶的都可行。然后我们构造出一个与 S 不同奇偶的最大的数 T，这个 T 的构造贪心就好了，找到最靠左和最靠右第一个不为 2 的数，算一下哪个比较优，然后求出 T。则与 S 同奇偶或者小于等于 T 的就都可行了。时间复杂度 $O(n)$。

2241: [SDOI2011]打地鼠

我们不难发现行和列其实是完全独立的两个操作，设最优答案的矩形是 $a\times b$ 的，那么我们每次如果只能操作一行的话，最大一定是操作 a 这么长，如果每次只能操作一列的话，最大一定是操作 b 这么长，因此我们可以分开来解决这两个问题。现在的问题就转变成一行数，每次可以把一个固定长度中每个数都减一，前提是每个数都正，问最后能否都减成 0，我们不难想到对这行数进行差分处理，然后就可以在 $O(n)$的时间内解决一行，所以解决一个矩阵就是 $O(nm)$，那么解决整道题就是 $O(nm(n+m))$。

2242: [SDOI2011]计算器

第一问快速幂，第二问扩展欧几里得，下面我们说一下第三问。题目所求为最小的 X 使得$Y^X\equiv Z\pmod P$，我们令 $M=\left\lceil\sqrt{P}\right\rceil$，$X=iM+j$，则$Y^{iM+j}\equiv Z\pmod P$，$Y^j\equiv\frac{Z}{Y^{iM}}\pmod P$，$Y^j\equiv Z\times Y^{iM(P-2)}\pmod P$，我们发现 i 和 j 的取值都是在 M 以内的，所以我们预处理同余方程的左边，然后用对 P 的余数做一个 HASH，然后我们枚举右面的 i，用之前预处理的 HASH 对应求出左边的 j，然后用 $iM+j$ 来更新答案即可。

2243: [SDOI2011]染色

来复习一下树链剖分，这道题比 ZJOI 的那道麻烦一下，需要给线段树打一个 lazy 标记，然后在需要下传的时候下传就可以了。至于维护一段数有多少段，就像 NOI2007 项链工厂那样做就可以了。

2245: [SDOI2011]工作安排

简单的费用流，设置附加源点 S，附加汇点 T。S 向每个员工连对应费用的边，容量为当前区间的长度，最后连一条容量为正无穷，费用为输入的最大费用的边。每个工作向 T 连容量为需求量，费用为 0 的边。每个员工向自己能做的工作连容量为正无穷，费用为 0 的边。求出费用流就的答案。

2257: [Jsoi2009]瓶子和燃料

题目所求为一个最大的 x，使得 x 至少是 n 个数中 k 个数的约数。

2272: [Usaco2011 Feb]Cowlphabet

这是一个简单的DP问题。首先根据信息进行预处理，$map[i][j]$表示字母 i 和字母 j 可以组合。定义状态 $dp[i][j][k]$ 表示前 $i+j$ 个字母中有 i 个大写字母，j 个小写字母，最后一个字母为 k 的方案数。则我们可以用 $dp[i][j][k]$来更新后继状态。如果 p 为大写字母则更新 $dp[i+1][j][p]$，否则更新 $dp[i][j+1][p]$，其中 k 和 p 可以组合。

2274: [Usaco2011]Generic Cow Protests [Neal Wu, 2010]

首先我们写出朴素的 DP 方程，$dp[i]$表示到达第 i 个数为止的方案总数，那么显然有 $dp[i]=\Sigma dp[j](0<=j<i$ 且 $sum[j+1...i]>=0)$，边界条件是 $dp[0]=1$。这么做的时间复杂度是 $O(n^2)$的，必须优化。我们记录所有数的前缀和，也就是说 $sum[i]=sum[i-1]+a[i]$。那么对于当前的状态 i，我们需要找的是所有的 $sum[j]$使得 $sum[j]<=sum[i]$。所以我们可以把整个数列的前缀和进行排序，然后维护以前对应的位置后用线段树维护就可以了。对于当前的状态 i，我们可以二分查找到满足要求最大的 $sum[j]$，计算完状态 i 后，将 $dp[i]$插入线段树中对应的位置，答案则是 $dp[n]$，时间复杂度为 $O(n\log n)$。

2276: [Poi2011]Temperature

正着考虑比较麻烦，我们不妨考虑确定一个终点 i，i 最远能往左移动多少。首先可以观察到位置 i 一定要选择最高的，因为往回走需要降的时候能降下来，但是如果选低了，需要升的时候就升不回去了。我们设第 i 个位置，给定的最小值为 $L[i]$，给定的最大值为 $R[i]$。首先我们置位置 i 选择的高度为 $R[i]$，然后往左扫描，如果 $R[i]$在 $L[i-1]$到 $R[i-1]$的范围内，则位置仍然是 $R[i]$，否则如果 $R[i]>R[i-1]$，则把 $R[i]$降到 $R[i-1]$，如果 $R[i]<L[i]$则停止，因为一定到头了。模拟 n 次以后取最优值即可，这样做时间复杂度为 $O(n^2)$。接下来我们考虑如何优化这个算法。我们观察到选定了 $R[i]$以后，我们要找第一个 j 使得 $R[j]<R[i]$或 $L[j]>R[i]$。如果 $R[j]<R[i]$，则点 i 的答案就是点 j 的答案加上 $i-j$，否则的话就只能是 $i-j$，这样我们只需要在最快的时间内找到这个 j。对于一个 i，如果 $L[i-1]$和 $R[i-1]$包含了 $R[i]$，则 $i-1$ 不是我们所求的 j，因此我们可以考虑从 i 往左找最长的一段满足这段任意一个位置都有 $L[k]$和 $R[k]$包含 $R[i]$。我们可以把这个拆成两部分，也就是对于任意的 $L[k]$有 $L[k]<=R[i]$，对于任意的 $R[k]$有 $R[k]>=R[i]$，这样如果我们能把这两部分分别找到，取一个最大值就可以了。这两个东西可以用单调栈来维护一些信息，在 $O(\log n)$的时间复杂度下找到，因此整个算法的时间复杂度就位 $O(n\log n)$。

2277: [Poi2011]Strongbox

题目说如果 x 可行，y 可行，则$(x+y) \bmod n$ 也可行，下面我们证明如果 x 可行，y 可行，则 $\gcd(x,y)$也可行：

不妨设：$ax+by\equiv\gcd(x,y)(\bmod n)$，其中$a\geq 0$且$b\geq 0$。

我们首先只考虑$ax+by=\gcd(x,y)$，由裴蜀定理知此方程必然存在一组整数解，如果这组整数解中的 a 和 b 均为非负，则直接满足了我们的需求。否则我们考虑两个数存在负数的情况，易知 a 和 b 不可能同时为负，不妨设 $a<0$，因为$ax+by\equiv\gcd(x,y)(\bmod n)$，因此 $a+kn$ 和 b 均是此同余方程的解，所以我们一定可以找到一个足够大的 k，使得把 a 加上 kn 以后，a 变成整数，因此我们就证明了如果 x 可行，y 可行，则 gcd(x,y)也可行。因此如果我们选定了一个基础数据 p，则 p，$2p$，$3p$，……，$kp=n$ 就都可行了。如果我们找到了最小的满足条件的 p，则可以取得答案的最大值为 n/p，接下来我们考虑如何找到最小的 p。

我们设第 i 次尝试的密码为 $a[i]$。我们设 $a[m]$的一个约数为 x，则只有当 n 能整除 x 时，x 才是合法的 p，所以我们把 $a[m]$改写成 gcd($a[m]$,n)。则 $a[m]$的约数都是可能的 p。因此我们的问题就变成了找到最小的 p 使得 p 是 $a[m]$的约数，且 $a[1..m-1]$都不能整除 p。然后我们把 $a[1..m-1]$改写成 gcd($a[1..m-1]$,$a[m]$)，这样改写是为了简便后面的运算，我们发现如果 $a[1..m-1]$是 p 的倍数，则 gcd($a[1..m-1]$,$a[m]$)也是 p 的倍数， 而如果 $a[1..m-1]$不是 p 的倍数，则 gcd($a[1..m-1]$,$a[m]$)也一定不是 p 的倍数。接下来我们用$O(\sqrt{n})$预处理出 $a[m]$的所有约数，然后最朴素的想法就是从小到大暴力枚举每个约数看是否可行，如果可行就输出 n/p 作为答案，但这样会超时。我们观察到当我们把 $a[1..m-1]$改写成 gcd($a[1..m-1]$,$a[m]$)之后，$a[1..m-1]$也一定是 $a[m]$的约数，而考虑到 $a[m]$的约数不会特别多，但是 m 却可能很大，所以 $a[1..m-1]$中将会有很大一部分是重复的，于是我们把 $a[1..m-1]$中重复的数字删掉，这样整个 a 数组的大小就变成了与 $a[m]$的约数个数同数量级的了，这样做的话也会超时。一个优化就是我们在检验枚举出来的p的时候，按照a数组中数的从大到小顺序进行枚举，一旦一个整除就跳出，事实证明这个优化相当有用，加上以后可以通过所有测试数据。

2278: [Poi2011]Garbage

我们发现题目的要求是一些边必须经过奇数次，一些边必须经过偶数次。其实我们可以把经过偶数次的边都删掉，因为这些边是完全可以不走的。剩下的问题就变成了给定一些连通块，要求每个连通块经过每条边奇数次，求一个方案，用欧拉回路解决就可以了，注意判一下无解的情况，还有题目要求输出简单环，所以只要把欧拉回路的路径拆开再输出就可以了。

2280: [Poi2011]Plot

首先二分答案，接下来考虑当前的值如何检验是否可行。这个贪心就可以了，我们从 1 开始，计算出一个 i 使得把第 1 个到第 i 个点构造一个最小圆覆盖，这个半径不超过当前检验的值且 i 最大，如果这个 i 也二分找到会非常慢，因为构造最小圆覆盖的时间复杂度与点数有关。因此我们可以考虑倍增的方式来求出 i。如果最后的段数小于等于题目要求的则可行，否则不可行。

2296: 【POJ Challenge】随机种子

这题纯属是在搞笑啊，我们只需构造一个 $A=1234567890999999$。然后对于输入的 n，输出 $A-A \bmod n$ 就可以了，特判一下 $n=0$，输出 -1。

2298: [HAOI2011]problem a

对于一个人他说有 a 个人比他分高，b 个人比他分低（如果 $a+b>n$ 的话直接扔掉，这肯定是假的），那么就有区间$[a+1,n-b]$内所有人的分数相同。我们找到这 n 个区间（或者因为排除了假的导致不足 n 个）。接下来的问题就变成找到最多的合法区间。所谓合法区间是指找到的这些区间的任意两个，要么完全相同，要么交集是空集。注意一个细节，对于一个区间$[L,R]$最多只能选 $R-L+1$ 次，因为这个区间只能容得下 $R-L+1$ 个人。然后 $dp[i]$表示右端点为 i 的区间最多能选多少个，转移很显然了。时间复杂度 $O(n)$或者 $O(n\log n)$。

2301: [HAOI2011]Problem b

经典问题，莫比乌斯反演解决。

2321: [BeiJing2011 集训]星器

我们发现选择位置(i,j)和(i,k)选择一个星星进行操作，释放的能量为 $k-j-1=(2k-2j-2)/2$，我们定义一个点(i,j)的势能为 i^2+j^2，则选择上面两个点的进行操作以后释放的能量就等于$(E_{初}-E_{末})/2$，由于势能的变化只与初末位置有关，因此我们可以考虑构造一个矩阵 $c[i][j]=a[i][j]-b[i][j]$，这样就得到了每个点星星数目的变化，则最终的答案就等于 $\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\left(i^2+j^2\right)\times c[i][j]}{2}$。

2324: [ZJOI2011]营救皮卡丘

用费用流做，把每个点 i 拆成 i 和 i'，设一个点 O 代表 0 号点。定义四元组$<a,b,c,d>$表示一条边从 a 到 b 容量是 c 费用是 d。我们依次建边$<S,O,k,0>$，$<O,i,1,dis[0][i]>$，$<i,T,1,0>$，$<i',j,1,dis[i][j]>$（其中 $i<j$），$<S,i',1,0>$。$dis[i][j]$表示的是 i 到 j 在不经过大于 $\max(i,j)$的点的情况下的最短路，然后求出最小费用就是答案。

2325: [ZJOI2011]道馆之战

用树链剖分维护就可以了，细节比较多。我们考虑询问 a 到 b 的最大值，有可能出现的情况就是在某一个地方走不下去了，所以在线段树中要建立相应的信息来维护这个，还有就是我们求出了 a 和 b 的 lca，那么我们需要从 a 走到 lca，再从 lca 走到 b，这两个方向是相反的。鉴于以上，我们在线段树中维护一下信息 $dis[0..1][0..1]$，$ansl[0..1]$，$ansr[0..1]$，其中 $dis[i][j]$表示左端点的 i 方格走到右端点的 j 方格的最长路，$ansl[i]$表示包含左端点的能向右延伸的最长路是多长，这些信息都可以很容易地维护。但是我们考虑到存在由 a 到 lca 和由 lca 到 b 两个过程，这两个过程方向是相反的，一个是向着深度减少方向进行，一个是向着深度增加方向进行，我们不妨在线段树的每个节点建立两个记录，把上面的那些最值倒过来维护一下就可以了。

2330: [SCOI2011]糖果

差分约束系统，我们简记三元组$<a,b,d>$为一条边起点是 a，终点是 b，长度是 d。设置超级起点 S，向每个人连边$<S,i,1>$，对于条件 1：$<a,b,0>$，$<b,a,0>$，对于条件 2：$<a,b,1>$，对于条件 3：$<b,a,0>$，对于条件 4：$<b,a,1>$，对于条件 5：$<a,b,0>$。求一遍最长路，设到点 i 的距离为 $dist[i]$，那么答案就是 $\Sigma dist[i]$ $(1<=i<=n)$。注意判一下如果有负环则无解，对于条件 2 和条件 4 如果 $a=b$ 则无解。

2423：[HAOI2010]最长公共子序列

第一问是经典问题就不说了，设第一个串为 a，第二个串为 b，$f[i][j]$第一个到 i，第二个到 j 的最长长度。第二问我们设 $g[i][j]$表示第一个到 i，第二个到 j 的方案数，那么有：
$g[i][0]=g[0][i]=1$，
当 $a[i]=b[j]$时；$g[i][j]=g[i-1][j-1]$, $g[i,j]+=f[i-1,j]$ (当 $f[i-1,j]=f[i,j]$)，$g[i,j]+=f[i,j-1]$ (当 $f[i,j-1]=f[i,j]$)；
当 $a[i]\neq b[j]$时，$g[i,j]+=f[i-1,j]$ (当 $f[i-1,j]=f[i,j]$)，$g[i,j]+=f[i,j-1]$ (当 $f[i,j-1]=f[i,j]$)，$g[i][j]-=g[i-1][j-1]$ (当 $f[i-1,j-1]=f[i,j]$)。
$f[n][m]$和 $g[n][m]$分别是第一问和第二问的答案，注意空间会爆，加滚动数组优化。

2429：[HAOI2006]聪明的猴子

题目所求有多少个猴子可以到达所有的点。我们求出 n 个点的最小瓶颈生成树，就是让生成树中最大的边最小，这个最大边就是用 Kruskal 求最小生成树的最后一条边，那么猴子跳跃距离大于等于这条边长的就都可行了。

2431：[HAOI2009]逆序对数列

DP 可以解决，状态 $dp[i][j]$表示前 i 个数已经放好，产生 j 对逆序对的方案数。放第 i 个数可以产生 0 个，1 个，……，$i-1$ 个逆序对，则有 $dp[i][j]=\Sigma dp[i-1][j-k]$，$0<=k<=i-1$ 且 $j-k>=0$。这样做的时间复杂度是 $O(nm^2)$。把那个求和用一个前缀和记录一下，时间复杂度变成 $O(nm)$，可以过了。

2438：[中山市选 2011]杀人游戏

首先用 Tarjan 缩点，求出新图中入度为 0 的点的个数。有一个特例就是如果原图中有不少于一个孤立的点，就是这个点不与任何点相连，则答案再加一。

2440：[中山市选 2011]完全平方数

我们二分答案，下面考虑如何计算小于等于 n 的数中，有多少个数不含有平方素数的因子。这里直接给出公式：

$$\sum_{i=1}^{n} m(i)\left\lfloor \frac{n}{i^2} \right\rfloor 。$$

其中 $m(i)$ 为莫比乌斯函数，从容斥原理的角度很容易解释这个公式，时间复杂度 $O\left(T\sqrt{n}\log n\right)$。

2441：[中山市选 2011]小 W 的问题

这道题看起来很简单，但是看了数据以后发现有横坐标一样的，有纵坐标一样的，有横纵坐标都一样的，所以不能再像 CTSC 2008 的图腾那样简单地用树状数组维护。既然无法按照横坐标的一个顺序做，就不妨考虑按照纵坐标的顺序做，但是不管怎么做都得先离散化一下，从接下来的描述中我们可以发现只需要离散化横坐标即可。我们发现这个 W 形，从第三个点分开左右两边是对称的，也就是说我们只需要求出两边的 V 形有多少个，乘在一起就是这个点上的 W 形个数，因此我们只考虑第 1,2,3 个点形成的 V 形个数，另一侧的同理。我们用一

条扫描线从 y 轴最下面扫描到最上面。考虑当前经过的点 $A(x,y)$，我们设 A 是第 3 个点，那么第 2 个点应该是在 A 的左下方，我们设这样的点组成的点集为 B，那么第 1 个点就得在 B 的左上方且纵坐标大于 A 的纵坐标，我们设这样的点组成的点集为 C。现在我们考虑如何计算相应的 B 对应的 C 的个数。我们不妨过点 A 作 y 轴的垂线，设这条垂线上面组成的为区域Ⅰ，下面组成的为区域Ⅱ。考虑一个区域Ⅱ中的点，我们设第 i 个点，对应的答案为 z_i，那么所有的答案就是 sigma(z_i)，这样就求出了 A 左边的 V 字形的个数。当经过点 A 时，区域Ⅱ中应插入点 A，并插入相应的答案，区域Ⅰ中删除点 A。我们发现区域Ⅰ中的操作只涉及单点操作和区间求和操作，所以区域Ⅰ用一个树状数组维护。而区域Ⅱ涉及一段区间的减一操作和增加一个点的操作，用线段树维护就可以了。这里注意的是我们把横坐标一样的都放在一起维护，一种特殊情况是有可能一个点还没有被插入，但在做区间减一操作时，却把它的值减一了，这里我们可以对线段树的每一个节点增设一个 *size* 域，表示横坐标为这个值的点有多少个，即可解决。对于第 3 个点右侧的 V 字形也用相同的方式维护。我们设第 i 个点左边有 $a[i]$个 V 字形，右边有 $b[i]$个 V 字形，那么最终答案就是 sigma($a[i]\times b[i]$)。

2442: [Usaco2011 Open]修剪草坪

我们反过来考虑这道题，$dp[i]$表示不选第 i 个，所需要的最小代价。那么我们要求的答案就是所有的总和减去最小的代价。我们把读入的那个限制长度加 1，并设为 m，意思是任意一段的当度不能大于等于 m。设第 i 个的收益为 $a[i]$，则有 $dp[i]=\min(dp[j])+a[i]$。$i-m<=j<=i-1$。单调队列维护就可以了。

2463: [中山市选 2009]谁能赢呢？

奇数输出 Bob，偶数输出 Alice，我也不知道为啥，找找规律就行了囧。

2490: Zombie’s Treasure Chest

两种物品至少有一种的购买次数不超过 $\sqrt{n}$，证明非常容易。然后暴力枚举购买次数小的那个。

2500: 幸福的道路

我们考虑先求出从 i 开始能走的最长路，这个可以通过两遍树形 DP 轻松求出。然后我们记 $dp[i]$表示两人路线的最后一天从点 i 出发，第一天最早可以从哪里出发。一个显然的观察就是对于任意的 $i<j$ 有 $dp[i]\leqslant dp[j]$。因此在求 $dp[i]$的时候，如果决策 j 不合法，则对于任意的 $i'>i$，决策 j 都不合法。我们可以考虑用两个单调队列分别维护以 i 为结尾的最大值和最小值，然后就可以在 $O(n)$的时间复杂度下解决此题。

2501: [usaco2010 Oct]Soda Machine

我们不难发现取得最优值的 x 一定是某一个区间的端点。我们可以按照区间左端点递增的顺序对区间进行排序，将所有的端点取出进行离散化处理。考虑当前的端点值 y，维护一个堆，将堆中最小值进行删除直到堆中最小值大于等于 y 停止删除，将区间的右端点插入堆直到区间的左端点大于 y 停止插入，用堆中的元素个数更新答案。

2525: [Poi2011]Dynamite

二分答案以后自底向上贪心就可以了，记录 $dp1[i]$表示距离点 i 最近的一个被选择的节点到点 i 的距离，$dp2[i]$表示距离点 i 最远的一个仍然没有被覆盖的节点到点 i 的距离，然后推一遍就可以了。时间复杂度 $O(n\log n)$。

2526: [Poi2011]Inspection

我们不妨假设 1 为根节点，设根节点的儿子中，子树大小的最大值为 S 则：

①．$S-(n-SS-1)>1$，无解。

②．$S-(n-SS-1)=1$，答案为每个点的深度和的二倍减去 S 对应的子树中的点的深度最大值。

③．$S-(n-SS-1)<1$，答案为每个点的深度和的二倍减去所有节点中深度最大的那个点的深度。

我们首先随便找一个节点作为根节点进行一次 DP，记 $down[i]$表示以 i 为根的子树的所有点到 i 的深度之和（根节点深度为 0），$size[i]$表示以 i 为根的子树的大小（包括 i），$fa[i]$表示 i 的父亲，$up[i]$表示 i 先走到 $fa[i]$，然后从 $fa[i]$开始走，走到所有非 i 的子树节点的深度和。$son[i]$表示 i 的儿子集合。则有：

$$size[i]=\sum_{j\in son[i]} size[j]+1,$$

$$down[i]=\sum_{j\in son[i]} down[j]+size[i]-1,$$

$$\begin{aligned} up[i] &= up[fa[i]]+n-size[fa[i]]+down[fa[i]]-down[i]-size[i]+size[fa[i]]-size[i] \\ &= up[fa[i]]+n+down[fa[i]]-down[i]-2\times size[i] \end{aligned}。$$

我们处理出这些信息以后，则 $up[i]+down[i]$ 就是把根节点设成 i 以后，所有点的深度之和。我们在 DP 的过程中求出每个点 $size$ 最大的儿子，以及每个点往下最长的链是多长，次长的链是多长，然后维护一些信息即可解决此题。时间复杂度 $O(n)$。

2527: [Poi2011]Meteors

首先根据读入的信息处理出第 i 个国家的位置集合，用个 $vector$ 或者链表存一下。然后用分治解决就可以了。我们分治陨石雨的位置，记当前等待处理的国家集合为 S，分治的陨石雨区间为$[L,R]$，我们取 $mid=(L+R)/2$，对于一次陨石雨，如果 $l<=r$ 我们可以看成是把 l 到 r 都加 d，否则可以看出把 l 到 m 都加 d，再把 1 到 r 都加 d。因此我们需要一个能支持区间加减，单点询问的数据结构，使用树状数组即可。我们首先处理$[L,mid]$的陨石雨，对于一个国家，如果当前获得的数量达到或者超过了它的需求，则把它分进 A 集合，同时把需求的数量也放进去，如果没达到要求，记需求为 x，当前获得为 y，则把它放入 B 集合，同时把 $x-y$ 作为新的需求数量也放入 B 集合。这样 A 集合中的答案就都在$[l,mid]$中，B 集合中的答案就都在$[mid+1,r]$中，递归处理两边即可。由于每个国家最多被处理 $\log k$ 次，因此时间复杂度最大为 $O(m\log m\log k)$。

2528: [Poi2011]Periodicity

我们可以把题目中描述的整数集合进行一下转化，记这个集合为 A，给定的英文字符串为 S，则对于任意的 $x\in A$，有 S 的开头 x 个字母组成的字符串和 S 的结尾 x 个字母组成的字符串相同。我们可以通过 HASH 的方式求出 A 集合中的所有元素，接下来我们考虑如何构造字典序最小的解。我们设集合 A 中的元素为 $p[i]$，按照 $p[i]$从小到大的顺序进行处理。对于一个 $p[i]$，如果 $p[i-1]\times 2>=p[i]$，则我们可以根据在 $p[i-1]$时确定的字符串来确定 $p[i]$范围内的字符串。否则假如 $p[i-1]$从起点开始确定的是字符串 B，则到达 $p[i]$时，我们需要确定的字符串为 B……B，因为这个串的末尾必须和前面相同，所以我们剩下的问题就是确定……是什么。这个……只能是 00000000..000，或者是 00000000..001。首先我们可以发现不管……填什么，这个 B……B 都是合法的，因此我们只要考虑字典序最小，还要考虑不存在一个 j 满足 $p[i-1]<j<p[i]$，使得 j 也满足了题目的条件，我们首先把……用 00000000..000 来代替，然后检验一下是否合法（所谓合法是指上面描述的 j 不满足题目的条

件），假设不合法了，那么一定是存在 $S000000=000000S$，则有 S 的某一位对应着最后一个 0，我们考虑把 00000000..000 的最后一位换成 1，不妨假设仍然不合法，则有 $S000000=000001S$，则 S 的那一位就对应 1 了，这时显然 $S000000\neq000001S$，因此如果 00000000..000 不合法，则 00000000..001 必然合法。由于只有当 $p[i]$超过 $p[i-1]$的二倍时我们才会去检验是否合法，因此最多检验 log$n$ 次，如果每次都用 $O(n)$的 HASH 来检验，则最坏时间复杂度为 $O(n\log n)$。

2529: [Poi2011]Sticks

我们把所有木棍按照长度从小到大进行排序。然后依次枚举每根木棍，记当前枚举的木棍作为三角形的最长边，我们维护三根颜色不同的木棍，使得这三根木棍是之前枚举过的最长的。然后用当前这根最长的去和那三根中的两根进行尝试，如果可行就输出，如果到最后也没找到解那就是无解了。因为如果选择最长，次长，次次长（这三根颜色不同）的都不合法，那么就一定无解了。

2530: [Poi2011]Party

这题真心乱搞，选一个点 a，找到一个不与 a 相邻的点 b，如果 b 没有被删除则同时删除 a 和 b，直到剩余的点变成 $n/3$，可以证明这样做是正确的。

2538: [Ctsc2000]公路巡逻

简单的 DP。状态 $dp[i][j]$时间 i 到达第 j 站最小相遇次数。$dp[i][j]=\min(dp[k][j-1]+getvalue(k,i,j-1))$。其中 $i-600<=k<=i-300$。$ans=\min(dp[i][n])$。

2539: [Ctsc2000]丘比特的烦恼

最开始题目没看清楚，他说“选择的人”指的是输入的所有人，也就是得让每个人都能找到一个伴侣，所以是求二分图最大权完备匹配，而不是二分图最大权匹配。用最大费用最大流就行了。

2542: [Ctsc2001]终极情报网

裸费用流，取个对数就可以加了，注意 SPFA 时候的精度问题。

2548: [Ctsc2002]灭鼠行动

没什么可说的，代码能力题，使劲写就行了。

2557: [Poi2011]Programming Contest

易知每个人最多能做 $\min\left(m,\left\lfloor\frac{t}{r}\right\rfloor\right)$道题，我们不妨把 t 先除以 r，这样做每道题的时间就变成了 1，然后再把

答案的时间乘 r。这个经典问题可以用费用流解决，但是会超时。注意到费用流也是基于如果一倍点没增广过，二倍点一定不可能增广到，因此我们可以可以考虑利用改进的匈牙利算法解决此题。首先记 $tt=\min\left(m,\left\lfloor\frac{t}{r}\right\rfloor\right)$，然后从 1 枚举到 tt，再从 1 枚举到 n，根据匈牙利算法的思想，对于打当前的 $i\in[1,tt]$，$j\in[1,n]$，如果增广第 $f(i,j)$ 个点失败，则 $f(i+k,j)$也会失败，因此如果一个点某此失败我们就可以永远地删掉它，这就大大减小了时间复杂度，可以通过所有的测试数据。

2561：最小生成树

我们考虑第一种情况，就是删多少条边能让一条边可能在最小生成树上。我们发现能影响这条边的一定是边权小于它的边，设这条边是由 S 到 T 的，我们找到所有边权小于这条边权的边，发现只要求出最小割即可，因为如果 S 到 T 有增广路的话那么这条边就一定不会在最小生成树上。而最大生成树与最小生成树涉及的边完全是对立的，所以我们可以用一样的方法再求一次最小割，两个结果加在一起就是答案了。

2563：阿狸和桃子的游戏

我们可以发现如果一条边的两个顶点是属于一个人的，那么他得到的将是两个点权加上边权，否则则是其中的一个点权。所以我们把每条边的边权平均分配到两个点上，这样就只剩下了点权，然后对所有的点权进行排序。那么先手取到的一定是第 1 大，第 3 大，第 5 大……的，而后手则是第 2 大，第 4 大，第 6 大的，扫描一遍看可以了。

2565：最长双回文串

由于有奇偶两种回文串，太麻烦了。做这样一个变换，在任意两个字母之间加入一个#，这样任意一个回文串都是以某一个字母为中心对称的串，我们用后缀数组或者 AC 自动机或者扩展 KMP 或者一个非常神的 $O(n)$的算法求出对于第 i 个字符，它的左右两端最多能扩展出多少个字符（不包括自身），设可以扩展出 $a[i]$个，那么以第 i 个字符为中心的回文串长度就是 $a[i]$。考虑两个位置 i 和 j，不妨设 $i<j$。我们假设这两个位置分别是两个回文串的中心。经过观察，我们可以发现，这两个串如果能连起来的话，需要满足 $a[i]+a[j]>=j-i$，这个自己画画就好了。此时两个两个回文串的总长度就是 $j-i$。也就是我们要最大化 $j-i$。我们发现有两个变量，很麻烦，我们对上面的不等式做这样一个变换，$a[i]+i>=j-a[j]$。我们不妨设不等号左面为 $A[i]$，不等号右面为 $B[i]$。接下来，我们把 $A[i]$和 $B[i]$都按照从小到大的顺序排序，然后扫一下这两个数组就可以求出结果了。

2580：[Usaco2012 Jan]Video Game

把所有的串建成一个 AC 自动机，状态 $dp[i][j]$表示第 i 个字符，在节点 j，能匹配的的最多个数，我们用 $dp[i][j]$去更新 $dp[i+1][?]$就可以了，边界条件 $dp[0][root]=0$，答案是 $\max(dp[i][j])$。

2582：[Usaco2012Jan]Bovine Alliance

水题，多个连通分量之间互不影响，考虑每个连通分量。设该连通分量有 n 个点 m 条边，如果 $m>n$ 直接判无解，输出 0。如果 $n=m$ 说明是一棵树上面长一个环，环上的点分配方案有两种，树上的分配方式唯一，所以总答案乘 2。如果 $m=n-1$，说明是一棵树，由于每个点都得带一条边，所以要选出 $n-1$ 个点带上边，剩下的

那个点不带边，也就是选出一个不带边的点，有 n 种选法，所以答案乘 n。

2584: [Wc2012]memory

我们首先考虑如何判断两个线段在移动的过程中会发生阻碍。由于四个方向是等价的，我们不妨只考虑向上移动的情况。设线段 A 的两点分别是(x_1,y_1)，(x_2,y_2)，线段 B 的两点分别是(x_3,y_3)，(x_4,y_4)，其中满足 $x_1<x_2$，$x_3<x_4$。我们先作出A和B在x轴上的公共投影，不妨设这个投影在x轴上的线段为$(L,0)$，$(R,0)$。我们不难发现$L=\max(x_1,x_3)$，$R=\min(x_2,x_4)$，如果按照上面计算出来的 $L>R$，那么显然移动过程中是不会发生阻碍的，因为在 x 轴上的投影为空，如果投影不为空，那么就有可能发生阻碍。我们设线段 A 所在直线为 $y_1=k_1x+b_1$，线段 B 所在直线为 $y_2=k_2x+b_2$，那么我们只要令 $x=L$，去算出对应的 y_1 和 y_2，如果 $y_1>y_2$，也就是说线段 A 在线段 B 的上方，那么在 B 向上移动的过程中会被 A 阻碍，否则不会。

我们不难发现将所有的线段都向上移动一定可以找到一个可行解，接下来我们可以考虑构造一个有向无环图，若线段 A 可以阻碍线段 B 的移动，则连一条有向边 A 到 B，那么我们只需要求出这个有向无环图的拓扑序列就得到了第二问的答案，但是这个边数将多达 n^2，是无法在规定时间内出解的。所以我们考虑简化这个有向无环图。我们发现若 A 限制 B，B 限制 C，那么我们是不需要连 A 到 C 的边的。因此我们把每个线段的两个端点变成两个事件，然后用一个垂直于 x 轴的扫描线从左到右进行扫描，对于一个线段，左端点为插入事件，右端点为删除事件，我们用一个平衡树来维护线段之间的关系。定义一种比较方式线段 A 大于线段 B 当且仅当在 B 向上移动的过程中会被A阻碍。我们可以发现对于所有插入到平衡树中的线段，它们在x轴上必然有长度大于 0 的公共投影，因此上面的比较方式是成立的。对于插入的线段 A，我们在平衡树中找到它的后继，由后继向 A 连一条边，找到它的前驱，由 A 向前驱连一条边，这样边数最多只有 $2n$ 条，每个线段插入一次删除一次，这样我们就在 $O(n\log n)$ 的时间复杂度下解决了第二问。

接下来我们考虑第一问，因为要求出最早的不合法移动，所以我们可以依次假设不合法的移动出现在上，右，下，左四个方向，求出这四个方向上不合法答案的最小值就是我们最终所求的答案了。我们以向上移动为例，剩下的三个方向可以通过将线段的端点以原点为中心进行顺时针 90° 旋转得到。我们考虑逆序处理所有的操作，假定我们已经处理完了第 $i+1$ 到第 n 个操作，当前处理的是第 i 个操作。我们把所有线段的端点进行离散化，接下来我们就要用到拓扑序列来解决。当扫描到一个线段(x_1,y_1)，(x_2,y_2)的时候，我们去查询 x 轴(x_1,x_2)上对应坐标拓扑序最小的点，记这个最小值为 a，线段 i 的在拓扑序列中的位置为 b，那么如果 $a<b$，也就是说在 i 移动的时候，由于$[i+1,n]$未移动，但是 i 的上面有一个线段，那么 i 必然被阻碍。处理完第 i 个线段，我们只需用第 i 条线段在拓扑序列中的位置来更新线段树$[x_1,x_2]$的最小值即可，这样第一问也在 $O(n\log n)$的时间复杂度下解决了。

2588: Spoj 10628. Count on a tree

把所有权值离散化以后树链剖分。对剖分后的重链建函数式线段树，由于已经离散化了，所以每个点的点权范围在$[1,n]$。对于一个询问$<a,b,k>$，我们二分第 k 小的点权是多少，设当前二分等待检验的值为 mid，我们要找到一个最小的 mid 使得小于等于 mid 的点权个数大于等于 k，那么当前 mid 离散化之前的点权就是答案了。

2591: [Usaco 2012 Feb]Nearby Cows

水题，按照层次遍历顺序推两遍就行了。

2594: [Wc2006]水管局长数据加强版

我们发现题目所求为求 a 到 b 的一条路径，使得这条路径上最长的边最短，并且要求支持随时删除一条边的操作。我们不难发现如果我们预处理出原图的一个最小生成树，那么树上 a 到 b 的路径就是满足题目要求的，这

个可以很容易地用反证法进行证明。由于删边这个操作比较麻烦，而且题目本身没有要求在线算法，所以我们考虑离线所有的操作，将所有的操作进行反转，这样一个删除操作就变成了一个插入操作。那么我们现在就需要一种数据结构支持添加一条边，删除一条边，查询两点间的最大边权。我们考虑一条边<*a*,*b*>边权为 ***v***，我们不妨新建节点 *c*，将边<*a*,*b*>拆成<*a*,*c*>和<*c*,*b*>，其中点 *c* 带上点权，权值为 ***v***，这样边权的问题就变成了点权的问题。我们考虑插入一条边，先查询 *a* 到 *b* 的最大点权，如果这个点权小于等于等待插入的点权我们则忽略这个插入操作，否则我们找到点权最大的那个点，删掉与它相邻的两个点，然后按照前面的做法把这条边和对应的权值维护好。整个过程用 LCT 就可以了。

2599: [IOI2011]Race

直接用树分治解决就可以了。考虑当前重心 *p*，有两种情况，第一种是路径经过 *p* 的，第二种是完全在 *p* 的某棵子树内。对于第二种情况可以递归处理，对于第一种情况，我们把 *p* 的子树中的所有点分成两部分，存在两个数组里，并记录到 *p* 的路径长度，到 *p* 的边数，然后对两个数组按照路径长度为关键字分别排序，最后归并两个数组更新答案即可，时间复杂度 $O(n\log^2 n)$。

2600: [Ioi2011]ricehub

我们考虑一个区间[*L*,*R*]，如何计算它的代价。显然米仓要建在中位数的位置上，然后代价就可以很简单地用前缀和计算。因此我们考虑枚举 *R*，随着 *R* 的增加，*L* 的最小值也是增加的，这个是单调的，只要线性扫描一次就可以了。时间复杂度 $O(n)$。

2620: [Usaco2012 Mar]Haybale Restacking

这题就像均分纸牌一样，首先我们根据数据预处理每个的差量是多少。然后记录一个前缀和，我们设前缀和一次为 *s*[1],*s*[2],……,*s*[*n*]，那么我们其实要找到的是一个 *k*，使得有 min(abs(*s*[*i*]－*s*[*k*]))，$1\leqslant i\leqslant n$。很容易想到把前缀和进行排序，取 $k=n/2$ 也就是中位数，证明是很显然的，然后算一遍就可以了。

2621: [Usaco2012 Mar]Cows in a Skyscraper

n 很小，爆搜就行。

2631: tree(伍一鸣)

直接用动态树解决就可以了。维护每个节点的两个 *lazy* 标记，*add* 和 *mul*，意义是每个点被加了多少，被乘了多少，这里需要注意的是标记下传的时候需要先传 *mul* 再传 *add*，然后就可以通过了。

2635: apex(卓亮)

这道题我的算法是 $O(\log^2 n)$的，卡了常数才通过。我们记 S_i 表示 *x* 的前 *i* 项的和。那么很容易证明出 $S_{4k}=2\times S_k$，$S_{4k+1}+S_{4k+2}=0$。并且对于 x_i 的计算我们可以在 $O(\log i)$的时间内完成。然后我们可以预处理 200W 以内的答案，对于这样的部分 $O(1)$回答就可以通过了。

2656: [Zjoi2012]数列(sequence)

对于给定的 n，首先把 n 不断除以 2，直到 n 变成奇数为止。开一个堆栈，除栈顶外，每层放两个数，栈顶放 n。记第 i 层放的是 $a[i]$和 $b[i]$，第 $i+1$ 层放第 i 层的奇数整除 2 和第 i 层的奇数整除 2 加 1，直到有一层的两个数为 0 和 1 为止。然后倒着计算就可以了，需要写个高精度。

2657: [Zjoi2012]旅游(journey)

共剖分出 $n-2$ 个三角形，若三角形 i 和三角形 j 之间有一条公共边，则由 i 向 j 连一条边。易证最终形成一棵 $n-2$ 个点的树，求出最长链就是答案。

2659: [Beijing wc2012]算不出的算式

当 $p=q$ 时，答案是$(p^2-1)/4$，否则答案是$(p-1)\times(q-1)/4$。

2660: [Beijing wc2012]最多的方案

记忆化搜索就行。

2661: [BeiJing wc2012]连连看

显然是一道二分图最优匹配问题，用费用流解决。建立附加源点 S，附加汇点 T。将$[L,R]$内的每个数拆成两个点，设 i 拆完后的点为 i 和 i'，如果 i 和 j 两个数满足题目的要求（也就是 i 大于 j，且 $i^2-j^2=k^2$，k 正整数，j 与 k 互质）那么就连一条 i 到 j'容量为 1，费用为 $i+j$ 的边，j 到 i'容量为 1，费用为 $i+j$ 的边。并且 S 向每个 i 连容量 1 费用 0 的边，每个 i'向 T 连容量 1 费用 0 的边。求出最小费用最大流，最大流除以 2 就是能配的对数，费用流除以 2 就是总和。

2663: [Beijing wc2012]灵魂宝石

二分答案就可以了，我们发现这是两个完全相反的问题。我们设当前被检验的值为 k，对于第一问，我们把所有距离小于等于 k 的连边，然后求出最大匹配，大于等于题目要求则可行；对于第二问，我们发现当 k 无穷大的时候，所有的都可以被匹配，也就是我们要把 k 从无穷大降下来，使得最大的那个刚换满足题目要求的 k 就是所求的 k，把距离大于等于 k 的连边，这表示这些不能进行匹配，然后求出最大匹配，这表示最多可以让多少对不能匹配，然后用 n 减去这个值如果小于等于题目要求则可行。

2668: [cqoi2012]交换棋子

拆点以后做费用流。我们设初始棋盘为 $A[i][j]$，目标棋盘为 $B[i][j]$，可以移动的次数为 $lim[i][j]$。我们假设所有 1 是有棋子，所有的 0 是空格。我们发现如果一个棋子在初始棋盘和目标棋盘中都出现那么这个棋子其实是没有用的，我们可以直接删除这样的棋子，因为它不会影响别的棋子的移动，证明很简单。如果 A 中的棋子个数和 B 中不相等，直接输出 -1。否则建立费用流，设置附加源点 S，附加汇点 T。此时对应的一个点(i,j)要么是 A

中有棋子，要么 B 中有棋子，要么 A 和 B 中都没有棋子。记一条边<a,b,c,d>表示从 a 到 b，容量是 c，费用是 d。S 向 A 中有棋子的连边<$S,A[i][j],1,0$>，B 中有棋子的向 T 连边<$B[i][j],T,1,0$>，若一个点(i,j)在 A 中有棋子或在 B 中有棋子，则连边<$A[i][j],B[i][j],(lim[i][j]+1)/2,0$>，否则连边<$A[i][j],B[i][j],lim[i][j]/2,0$>，对于两个点$(a,b)$，$(c,d)$。如果两个点可以相互到达，则连边$(A[a][b],B[c][d],+\infty,1)$。求出最小费用流，同时记录最大流，如果最大流等于 A 中的棋子个数则输出费用流，否则输出－1。

2678: [Usaco2012]Bookshelf

我们设高度为 $a[i]$，宽度为 $b[i]$，单层宽度限制为 L。一个朴素的 DP 方程是 $dp[i]$表示处理完前 i 本所需要的最小值，则有 $dp[i]=\min\limits_{j=f[i]}^{i-1}\left(dp[j]+\max\limits_{k=j+1}^{i}(a[i])\right)$，$dp[1]=a[1]$，其中 $f[i]$表示满足当前层宽度不超过 L 的最小的 j。直接做的时间复杂度为 $O(n^3)$，观察发现如果将 j 倒序枚举，那么括号中求最大值的部分可以通过 $j+1$ 在 $O(1)$的时间内得到更新，时间复杂度变为 $O(n^2)$。进一步观察我们可以发现可能在最优决策中出现的 j 只有两种情况，第一种是 $j=f[i]$，第二种是将所有这样的 j 连起来，$a[j]$是一个单调递减的序列。对于第一种情况不难理解，我们来证明第二种情况的正确性。考虑当前决策的最后一本书为 p，以及 i,j,k 为三本书，满足 $i<j<k<p$，$a[j]<a[k]<a[i]$。则我们可以发现这三个数中 i 是最大的，k 是最小的，j 介于 i 和 k 之间，那么选 k 的话一定不如选 i 优，理由是 $i+1$ 到 p 的最大值一定不会比 $j+1$ 到 p 的最大值大，因为存在 $a[k]>a[j]$，且我们不难观察出 $dp[i]<=dp[j]$，因为 $i<j$，所以对于这样的 j 我们可以扔掉。具体做法是维护一个单调队列，记录一个 $a[i]$递减的序列。但是下一个问题就是队首元素一定的最优吗？显然答案是否定的。所以我们可以采用平衡树的方式维护这个单调队列，记 i 和 j 为单调队列中相邻的两个元素，那么显然由 i 计算得到的值为 $dp[i]+a[j]$。因此在维护当前书本的过程中，我们首先保证决策的合法性，对队首元素进行删除，直到队首元素大于等于 $f[i]$，接下来我们对 i 进行决策，最后我们把 i 插入单调队列，进行队尾删除以保证元素的单调性。对于区间最小值的查询可以用 RMQ 在 $O(n\log n)$预处理，$O(1)$查询的时间复杂度解决。整个算法的时间复杂度为 $O(n\log n)$。

2679: [Usaco2012]Balanced Cow Subsets

由于 n 最大只有 20，我们发现朴素的搜索方式是 $O(3^n)$，因为每个元素只有三种情况：不选，加入 A 集合，加入 B 集合。但是这显然无法通过所有测试数据。我们可以把 n 一分为二，这样每一份最多就只有 10 个，对每一份进行搜索，这样时间复杂度变为 $O(3^{n/2})$，最后我们对两部分进行合并即可。

2705: [SDOI2012]Longge 的问题

$$\sum_{i=1}^{n}\gcd(i,n)=\sum_{i|n}i\times j\left(\frac{n}{i}\right)。$$

2708: [Violet 1]木偶

DP 就可以了，状态 $dp[i]$表示前 i 个最多失配多少个，$dp[i]=\max(dp[j]+f[j+1][i])$，其中 $f[i][j]$表示 i 到 j 最多失配多少个，比如我们检验 i 到 j，失配 k 个，那么我们显然是让 i 到 $j-k$ 去匹配 $i+k$ 到 j。时间复杂度 $O(n^4)$。

2709: [Violet 1]迷宫花园

水题，二分答案以后用最短路判断是否可行。但是读入太恶心了，我调了好长时间。

2711: [Violet 2]After 17

我们不难发现横纵坐标是两个独立的过程，我们可以分开考虑。以横坐标为例，题目所求为 $\sum x_i x_j\,(i<j)$ 的最小值，这个值可以写成 $\frac{\left(\sum_{i=1}^{n} x_i\right)^2-\sum_{i=1}^{n} x_i^2}{2}$，我们发现一次旅行要么是走到最左面要么是走到最右面，因为这是个线性的关系。所以对于每个每个 x_i，要么是题里输入的值，要么是它的相反数。因此我们只需要最小化 $\left(\sum_{i=1}^{n} x_i\right)^2$，也就是最小化 $\left|\sum_{i=1}^{n} x_i\right|$，由于每个 x_i 都不是很大，所以可以用一个背包 DP 来解决，$dp[i][j]$ 表示前 i 个能否取到 j，转移就很显然了，不写了。

2712: [Violet 2]棒球

做法参见杨哲论文《一类分数问题的研究》。

2715: [Violet 3]最优指令

直接求最短路就行了。用二进制表示 S 集合中的元素，记录一下转移的位置。

2716: [Violet 3]天使玩偶

我们发现如果没有加点这个操作的话，就可以对所有的询问进行离线。然后按照 x 为关键字进行排序，按照 x 升序处理每一个询问，用树状数组就可以了，这个很简单，不多说了。接下来我们考虑有加点的操作，受到 CDQ Cash 一题神分治的启发，我们不难想到用类似的方法进行处理。定义过程 $Solve(l,r)$ 表示处理第 l 到第 r 个操作，我们设初始点有 n 个，操作有 m 个，定义 $N=n+m$，那么我们所需要的就是 $Solve(1,N)$ 的结果。对于 $Solve(l,r)$，我们可以递归 $Solve(l,mid)$，$Solve(mid+1,r)$，其中 $mid=(l+r)/2$。那么我们分别处理了上面的两个操作以后，只需要用 l 到 mid 中的结果去更新 $mid+1$ 到 r 的结果，这样便保证了结果的正确性。这个更新可以用类似归并排序的方式来实现，按照 x 为第一关键字 y 为第二关键字做一下就可以了。时间复杂度 $O(N\times\log N\times\log Len)$，其中 Len 为最大的坐标。

2717: [Violet 4]迷路的兔子

这是结论题，去网上看题解吧。

2720: [Violet 5]列队春游

我们将身高从小到大排序，把身高相同的分成一组，共分了 m 组，每组的左端点是 $l[i]$，右端点是 $r[i]$。我们记前 i 组一共有 $c[i]$个人，那么答案就是 $\sum_{i=1}^{m}\frac{(n+1)\times(r[i]-l[i]+1)}{n+1-c[i-1]}$。这个算式的证明我也不清楚，大致算算 $i=1$，$i=2$，$i=m-1$ 和 $i=m$ 发现都是对的，所以就当它是对的吧。

2721: [Violet 5]樱花

题目所求为，不定方程 $\frac{1}{x}+\frac{1}{y}=\frac{1}{n!}$ 的个数。我们整理一下，$x=\frac{yn!}{y-n!}$，我们不妨设 $y=kn!$，其中 $k\in R$，我们继续表示 k，令 $k=\frac{b}{a}$，其中 a 和 b 互质。我们带入上面的算式 $x=\frac{\frac{b}{a}n!n!}{\frac{b}{a}n!-n!}$，也就是 $x=\frac{b}{b-a}n!$，那么我们不难发现 $a\mid n!$ 且 $(b-a)\mid n!$。那么原问题就变成了求 $n!$ 中的两个约数，它们的最大公约数是 1。我们首先把 $n!$ 分解质因数，设第 i 个质数有 $f[i]$ 个。我们定义 $dp[i]$ 表示前 i 个素数有多少种方案，则有 $dp[i]=dp[i-1]\times(2f[i]+1)$。我们设共有 m 质数，则答案就是 $dp[m]$。

2722: [Violet 5]爱的花环

矩阵处理起来挺麻烦的，我们进行如下转化。首先我们设出现次数矩阵为 $P[i][j]$，最小值和最大值分别为 $L[i][j]$ 和 $R[i][j]$。我们令 $Q[i][j]=\begin{cases}P[i][j]+P[j][i] & i\le j\\ 0 & i>j\end{cases}$，$B[i][j]=\begin{cases}A[i][j] & i=j\\ 2\times A[i][j] & i<j\\ 0 & i>j\end{cases}$。那么原问题就转化成了让 $\frac{\sum Q[i][j]\times B[i][j](1\le i\le j\le n)}{2}$ 取得最小值。分母是定值，不用管了，只需要最小化分子。我们发现此时我们只需要在满足 $\sum B[i][j]=0$ 的条件下，最小化上面的值即可。此时已经与矩阵毫无关系，我们把 Q 和 B 两个数组拿出来，换成一个一维数组，每个值分别映射成 $L[i]$，$R[i]$，$X[i]$。那么我们只需要求出在满足 $L[i]\le X[i]\le R[i]$ 条件下，$\sum_{i=1}^{\frac{n\times(n+1)}{2}}P[i]\times X[i]$ 的最小值。下一步我们按照 $P[i]$进行排序，一个贪心策略是选定一个 k，使得 1 到 $k-1$ 都取 $L[i]$，$k+1$ 到 $\frac{n\times(n+1)}{2}$ 都去 $R[i]$，然后计算出 k 的取值，看是否可行，也就是是否在 $L[k]$和 $R[k]$的范围内。这里注意一个问题由于当第 k 个点不为主对角线上的点时，$B[i][j]=2\times A[i][j]$，所以此时 $B[i][j]$必须为偶数。如果第 k 个点不为主对角线且通过计算得到的第 k 个值为奇数的话，那么我们需要调整。调整的策略就是取 1 到 $k-1$ 一个主对角线上的点加 1 或者取 $k+1$ 到 $\frac{n\times(n+1)}{2}$ 一个主对角线上的点减 1，然后更新答案，注意一些细节就可以，比如第 k 个值选 $L[k]$的时候，不能决策减 1，取 $R[k]$的时候不能决策加 1 等等。

2724: [Violet 6]蒲公英

我们把原序列离散化，这样每个数的最大值也不超过 n。将数列分成 c 段，定义 $sum[i][j][k]$表第 i 段到第 j 段，值为 k 的数有多少个。我们求出这个以后再求出 $best[i][j][0]$，$best[i][j][1]$，表示第 i 段到第 j 段，最大的次数是多少，出现最多的数是多少。对于一个询问$[L,R]$，我们把它分解成$[L,l)$，$[l,r]$，$(r,R]$，其中$[l,r]$这部分可以在预处理的数组中处理完成，$[L,l)$和$(r,R]$加起来也没有多少个数，暴力查一下。这样做的预处理时间复杂度是 $O(c^2n)$，询问一次的时间复杂度是 $O(n/c+c)$，我们取 $c=n^{1/3}$ 即可权衡预处理和询问，不至于其中一个过慢。

2729: [HNOI2012]排队

首先不管两个老师相不相邻，直接把男生加上两个老师固定然后女生插空有$(n+2)!\times m!\times C(n+3,m)$种方案，然后考虑两个老师必须相邻的情况有 $2(n+1)!\times m!\times C(n+2,m)$种方案。则最终答案就是第一部分减掉第二部分，写个高精度就可以了。

2730: [HNOI2012]矿场搭建

首先求出原图的割点，然后我们把割点删掉，这样图就变成了一些连通块。我们发现如果要设置最少个数那么一定不会在割点上设置，如果一个连通块与两个割点相连，那么这个连通块不用设置。这样我们就只需要求出满足只与一个割点相连的连通块个数，就得到了第一问的答案，然后根据乘法原理把这样连通块中的点数乘起来就得到了第二问的答案。特判当原图没有割点的时候，第一问的答案是 2，第二问的答案是 $n\times(n-1)/2$。

2733: [HNOI2012]永无乡

用 SBT 启发式合并，可以使合并节点个数最多为 n 个，然后用并查集维护连通性。

2734: [HNOI2012]集合选数

我们考虑建这样的一张表，假定有了一个初始的数 x。其中 $a[1][1]=x$，$a[i][1]=a[i-1][1]\times 3$，$a[i][j]=a[i][j-1]\times 2$。那么原问题就转换成，在这张表中任意选择一些数使得任意两个数都没有公共边，由于表格很小可以用状压 DP 解决。x 的取值的是$[1,n]$内既不能被 2 整除也不能被 3 整除的数。最后把每张表的答案乘起来就是最后的答案了。

2738: 矩阵乘法（梁 盾）

由于题目没有要求在线，直接读入所有询问，然后分治所有询问即可，这个做法太经典了不多说了，用一个二维树状数组维护即可。时间复杂度 $O((n^2+Q)\log n\log Ans)$。

2741: 【FOTILE 模拟赛】L

我们发现，如果我们记录前缀的异或和 $b[i]$，那么 $a[l]\ xor\ a[l+1]\ xor\ a[l+2]\ \ldots\ xor\ a[r]$就变成了 $b[r]\ xor\ b[l-1]$。因此对于题目给定的一个询问$[L,R]$，我们就是要求 $L-1<=i<=j<=R$，$b[i]\ xor\ b[j]$的最大值。考虑将原序列分成 $\sqrt{n}$ 块，记录 $max[i][j]$表示从第 i 块的起点到 b 数组的第 j 个数组成的序列中，任选两个的异或最大值。显然 $max[i][j]$可以由 $max[i][j-1]$和 $b[j]\ xor\ b[k]$转移而来（其中 k 在第 i 块的起点到 j 组成的区间内），这个用字典

树维护就可以了。对于一个询问[*L*,*R*]，我们找到 *L* 属于哪个块，记这个端点为 *p*，则[*p*,*R*]的答案已知，接下来我们要求的就是对于[*L*,*p*)中的某个数，找到[*L*,*R*]的一个数，使得这俩数的异或和最大，这个可以通过函数式字典树解决。

2746：[HEOI2012]旅行问题

我们把输入的所有字符串建成一个 AC 自动机。对于询问其实就是求两个字符串的两个前缀，它们的最长公共后缀是多长，且满足这个最长公共后缀是从根节点到某一个节点的一条路径。根据 *fail* 指针的定义，我们可以发现从一个字符沿着它的 *fail* 指针走，那么每一条路都和它上一条路有一定长度的公共后缀，并且当前的这条路径一定可以从根节点走下来。我们按照 *fail* 指针建成一棵 *fail* 树，就是 *i* 的 *fail* 连一条向 *i* 的边。那么每个询问我们先找到这两个节点的位置，继续观察可以发现，这两个点在 *fail* 树上的 LCA 到根节点的路径所组成的字符串，一定就是题目所求的，因为这个字符串满足是这两个子串的公共后缀，并且这个字符串包含根节点，而且不存在比这个字符串更长的字符串了。

2748：[HAOI2012]音量调节

由于限制了最大音量只有 1000，所以我们可以考虑用背包 DP 解决。状态 *dp*[*i*][*j*]表示到第 *i* 首歌曲的时候，能不能取到 *j* 状态，设第 *i* 首需要的调节量为 *a*[*i*]，那么当 *dp*[*i*－1][*j*－*a*[*i*]]和 *dp*[*i*－1][*j*＋*a*[*i*]]有一个为真时，*dp*[*i*][*j*]就为真，边界条件是 *dp*[0][*begin*]＝*true*。

2751：[HAOI2012]容易题(easy)

我们设数的范围是 1 到 *n*，那么如果没有限制的话每个数可以取 1 到 *n*，$S=\sum_{i=1}^{n} i=\frac{n(n+1)}{2}$，考虑有些数有限制，我们设有限制的数共 *p* 个，其中第 *i* 个减去限制的所有数变成 *a*[*i*]，那么答案就是 $S^{m-p}\times\prod_{i=1}^{p} a[i]$。注意对输入数据的去重，注意及时取模，否则会出现两个 64 位整数的相乘。

2752：[HAOI2012]高速公路(road)

区间问题考虑用线段树解决。我们先不考虑修改操作。设原序列每个数为 *a*[*i*]，对于一个询问[*L*,*R*]，我们发现所求结果就是 $\frac{\sum_{i=L}^{R-1}\sum_{j=i}^{R-1}\sum_{k=i}^{j} a[k]}{C(R-L+1,2)}$。我们不妨只考虑分子，且[*L*,*R*]＝[1,*n*＋1]，至于如何求[1,*n*]的分子上面的那个式子，我们不妨设每个数被加权了 *b*[*i*]次，易知 $b[i]=i\times(n-i+1)$，则分子的值为 $\sum_{i=1}^{n} a[i]\times b[i]$。显然要想用线段树做这个问题必须要满足区间加法，而显然当 *n*＝1 时，*b*[1]＝1，所以我们只需要考虑如何把两个区间合并。我们考虑两个区间 *A*，*B*，对于一个区间[1,*n*]，我们令：

size＝*n*，

sum＝$\sum_{i=1}^{n} a[i]$，

$sumL=\sum_{i=1}^{n} i\times a[i]$，

$sumR=\sum_{i=1}^{n}(n-i+1)\times a[i]$，

ans 为这个区间的答案，则对于两个区间合并后的大区间有：

$size=size_A+size_B$，

$sum=sum_A+sum_B$，

$sumL=sumL_A+sumL_B+size_A\times sum_B$，

$sumR=sumR_A+sumR_B+size_B\times sum_A$，

$ans=ans_A+ans_B+sumL_A\times size_B+sumR_B\times size_A$。

接下来我们考虑修改操作，对区间[1,*n*]加 *d*，则：

$sum=sum+d\times size$，

$sumL=sumL+d\times\frac{size\times(size+1)}{2}$，

$sumR=sumR+d\times\frac{size\times(size+1)}{2}$，

$ans=ans+d\times\frac{size\times(size+1)\times(size+2)}{6}$，

至于推倒过程在这里就不写了。

2753：[SCOI2012]滑雪与时间胶囊

我们设数据中的边为<*a*,*b*>权值为 *d*，节点高度为 *h*[*i*]。虽然题目说是无向图，但是对于两个点之间的路线，如果两个点的高度不相等，那么只能从高的走到低的，否则可以双向走。第一问很简单，BFS 或者 DFS 一次就可以求出。对于第二问我们发现实际是求最小树形图，但是数据过大，无法在规定时间内跑出来。稍微观察一下，可以得到这样一个算法，我们把所有的边进行处理，使得有<*a*,*b*>中 *h*[*a*]>=*h*[*b*]。那么我们发现只要按照所有边的 *h*[*b*]为第一关键字降序，权值 *d* 为第二关键字增序排序后，用 Kruskal 算法得到的一个 MST 就是原图的一个最小树形图。

2765：[JLOI2010]铁人双项比赛

设总路程为 *L*，每个人的长跑和骑车速度分别为 $v_0[i]$和 $v_1[i]$，我们假设长跑路程为 *x*，则第 *i* 个人所需时间为$T_i=\frac{x}{v_0[i]}+\frac{S-x}{v_1[i]}$，整理得$T_i=\left(\frac{1}{v_0[i]}-\frac{1}{v_1[i]}\right)x+\frac{S}{v_1[i]}$，我们可以把 T_i 看成是斜率为$\frac{1}{v_0[i]}-\frac{1}{v_1[i]}$，截距为$\frac{S}{v_1[i]}$，定义域在[0,*L*]内的一条线段。不难发现最优值一定取在两条线段的交点上，我们可以用 $O(n^2)$的时间枚举交点，再用 $O(n)$的时间判断，但是要特判 *x*=0 和 *x*=*L* 两个特殊点。时间复杂度 $O(n^3)$。

2766：[JLOI2010]世界杯租房

我们预处理 $t[i][j]$表示第 i 天到第 j 天存在连续空房的字母最小的房间编号。记 $dp[i]$表示第 l 天到第 i 天安排好住处所需的换房次数。则 $dp[i]=\min(dp[j])+1$，其中 $t[j+1][i]$存在且 $j<i$。边界条件是 $dp[l-1]=0$。然后方案在 DP 的时候记录一下就好了。

2767: [JLOI2010]足彩投注

我们发现对于每场比赛的下注方式只有三种。第一种是压 0,1,2 的其中一个，第二种是压 0,1,2 的其中两个，第三种是把整个都压上。记 $s_{i,j}=\frac{p_{i,j}}{q_{i,j}}$，$b[i][0]=\max\left(s_{i,j}\right)$，$b[i][1]=\max\left(s_{i,j}+s_{i,k}\right)$，$b[i][2]=s_{i,0}+s_{i,1}+s_{i,2}$。则我们所求的结果就为 $\frac{M}{N}\times\max\left(\prod_{i=1}^{n}b[i][j]\right)$。设 $b[i][j]$中有 x 个 $b[i][1]$，有 y 个 $b[i][2]$，则根据题目要求我们需要满足 $2^x3^y\le U$。考虑使用 DP 解决。状态 $dp[i][j][k]$表示前 i 场比赛有 j 场选择了 $b[i][1]$，有 k 场选择了 $b[i][2]$所得到的最大期望值。则 $dp[i][j][k]$可以由 $dp[i-1][j][k]\times b[i][0]$，$dp[i-1][j-1][k]\times b[i][1]$，$dp[i-1][j][k-1]\times b[i][2]$三种方式转移得到。考虑到答案可能非常大，而且是求对数，因此我们不妨考虑整个过程都使用对数计算，这样所有的乘法就都变成了加法。则上述的转移方程变为在 $dp[i-1][j][k]+\ln(b[i][0])$，$dp[i-1][j-1][k]+\ln(b[i][1])$，$dp[i-1][j][k-1]+\ln(b[i][2])$。答案在 $dp[n][i][j]$满足 $2^i3^j\le U$ 的里面选一个最大值，再加上 $\ln\frac{M}{N}$ 即可，注意到 $U\le10^4$ 这个条件，显然 DP 数组中第二维的取值不超过 13，第三维的取值不超过 8，因此复杂度可以接受。

2780: [Spoj]8093 Sevenk Love Oimaster

把初始的 n 个字符串连在一起，相邻两个中间用一个特殊的字符隔开，然后跑一遍后缀数组，记录后缀的排名，以及第 i 个后缀的第一个字母属于哪个字符串。然后我们离线所有询问，对于读入的第 i 个询问，我们发现如果它可能在上面的字符串中出现的话，那么必须得是一些后缀的前缀和这个字符串完全匹配，由于后缀是排好序的，不难发现如果满足的话一定是连续的一段后缀，我们对这个询问的字符串进行扫描，初始时 $L=1$，$R=n$，每当遇到一个字符通过二分查找的方式来调大 L，调小 R，那么如果答案不为 0 的话，当扫描完成这个字符串的最后一个字母的时候一定可以可以得到一个区间$[L,R]$，那么后缀排名在这个区间的所有后缀的前缀都能完全和这个询问的字符串匹配。由于我们已经处理好了每个后缀的第一个字符所属的是哪个字符串，那么接下来的问题就变成了给定一些区间$[L,R]$，每次询问这个区间中有多少个不同的数。由于我们的做法是离线的，我们可以把所有的区间按照左端点为关键字排序，然后建一个树状数组扫一遍就行了，这个很经典，不多说了。

2786: Ural1142 Relation

可以发现等号连接的部分等价，记 $dp[i][j]$表示 i 个符号，共有 j 个不等价的部分的方案数（比如 10 个符号，填了 1 个等号，那么就有 9 个不等价的部分），则有 $dp[i][j]=dp[i-1][j]\times j+dp[i-1][j-1]\times j$，其中 $dp[i-1][j]$表示添加一个等号，把第 i 个符号放到 j 个部分中的一个，$dp[i-1][j-1]$表示添加一个小于号，前面的 $j-1$ 部分共有 j 种填法。记 $ans[i]$表示 i 个符号的最终答案，则 $ans[i]=dp[i][1\ldots i]$，边界条件是 $dp[1][1]=1$，$dp[i][0]=0$，写个高精度就行了。

2788: [Poi2012]Festival

对于每个 A 条件，我们连边 a 到 b 边权为 1，b 到 a 边权为 0，对于每个 B 条件我们连边 b 到 a 边权为 0。

对这个有向图进行强连通缩点，缩点以后容易发现不同连通分量之间互不影响。因此我们只需要考虑把每个连通分量的最大值加起来就是最后的答案了。对于每个连通分量的最大值就是这个连通分量内两两点最短路的最大值，这个最短路指的是缩点之前的图的最短路。证明非常的神奇，我也不知道是为什么，画几个看看可以发现确实是对的。

2789: [Poi2012]Letters

举一个例子，A＝“ABAAC”，B＝“CAABA”。我们把 A 编号为 1~n，也就是 12345，B 中为 A 中的对应序号，也就是 51324，那么显然 A 的编号序列没有逆序对，对 B 做一次变换可以使逆序对减一，那么 B 中逆序对的个数就是答案了。

2790: [Poi2012]Distance

我们设 $D(a,b)$表示由 a 变为 b 的次数，我们设 d＝gcd(a,b)，则 $D(a,b)$＝$F(a/d)$＋$F(b/d)$，其中 $F(x)$表示 x 的质因数个数，就是分解质因数以后所有指数加起来。我们设 $G(x)$为通过除法得到 x 的最小及次小的次数，然后用 $F(k/x)$去更新 $G(x)$即可，在这个过程中更新对 a 最小的 $D(a,b)$即可。

2791: [Poi2012]Rendezvous

我们发现原图其实是一个环带着树的结构，树是内向树。对于询问，分三种情况：

①.不在一个连通分量内的，直接输出两个－1。

②.在一个联通分量，且在同一棵子树，找到 LCA 然后输出 LCA 到询问点的距离即可。

③.在一个联通分量，不在同一棵子树，首先两个点走到环上，然后判是 a 到 b 更优还是 b 到 a 更优即可。

需要维护每个点所在子树的根节点，所在第几个环，环中节点距离，环中节点标号，每个点的倍增父亲，每个点的深度。时间复杂度 $O((n+m)\log n)$。

2792: [Poi2012]Well

我们二分平滑程度 z，我们先不考虑 $a[i]$＝0 这个东西，先考虑如何把整个序列平滑度变成 z，有一个比较神的贪心，代码只有两行：

for (i＝1;i<n;i＋＋) *if* ($a[i+1]$>$a[i]$＋z) $a[i+1]$＝$a[i]$＋z;

for (i＝n;i>1;i－－) *if* ($a[i-1]$>$a[i]$＋z) $a[i-1]$＝$a[i]$＋z;

合法性显然，最优性也可以证明但是我不会。

接下来我们考虑把一个 $a[i]$变成 0，如果 $a[i+1]$和 $a[i]$的差大于 z 了，我们则需要把 $a[i+1]$变成 z，依次类推，直到找到一个位置 p，$a[p]$和 $a[p-1]$的差不大于 z 为止，这样就处理完了 i 的右侧，i 的左侧也是同理。这样做每次检验的时间复杂度就变成了 $O(n^2)$。我们考虑当前的位置 i，往右最多能影响到 p，往左最多能影响到 q，有一个结论就是随着 i 的增加，p 和 q 都在单调增加，这点很容易证明。因此枚举 p 和 q 的位置就变成了每轮 $O(n)$，均摊到每个位置 i 上就变成了 $O(1)$。这样一轮检验的时间复杂度就变成了 $O(n)$，记最大的数为 m，则总时间复杂度为 $O(n\log m)$。

2793: [Poi2012]Vouchers

这道题暴力就可以了，可以证明时间复杂度为 $O(n\log n)$，其中 n 为最大的幸运数。需要做的一个优化就是记

录 *start*[*i*]表示枚举 *i* 的倍数时第一个枚举的数是多少，初始情况 *start*[*i*]＝*i*。

2794: [Poi2012]Cloakroom

离线所有询问，把物品按照 *a* 为关键字排序，询问按照 *m* 为关键字排序。然后两个指针扫扫就行了，状态 *dp*[*i*]表示满足总和为 *i* 的情况下，*b* 最小的物品的 *b* 最大是多少。转移方程 *dp*[*i*]＝max(*dp*[*i*],min(*b*,*dp*[*j*－*c*])，两路归并的好处就是每个物品只做了一次背包。时间复杂度 *O*(*n*log*n*＋*m*log*m*＋*nk*)。

2795: [Poi2012]A Horrible Poem

我们首先建立后缀数组，下面考虑[*L*,*R*]在 *T* 为长度的循环节是否可行，不难发现当 $LCP(L,T)\geq R-L+1-T$ 时可行，但是如果我们花 $O\left(\sqrt{R-L+1}\right)$ 的时间去枚举是无法通过所有测试数据的，必须换一种方法。继续观察不难发现，我们只需要把整个子串分为尽量多的部分，我们发现，若分为 *x* 份可行，则 *x* 的约数都可行。所以我们可以尝试求出最大的 *x*，具体做法是用预处理 2 到 710 的素数（710^2>＝500000），接下来我们把 $R-L+1$ 分解质因数，对于不同的素数因子，我们去尝试最大的次幂即可。最后 $\frac{R-L+1}{x}$ 就是答案。

2796: [Poi2012]Fibonacci Representation

记忆化搜索就可以了，预处理 *Fibonacci* 的第 1 到第 90 项，如果是 *Fibonacci* 数就返回 1，如果这个数小于等于 1000***W*** 则把这个数对应的答案记下了，记 *f*[*i*]为 *Fibonacci* 数的第 *i* 项，当前要找的数为 *x*，我们找到最小的 *f*[*i*]使得 *f*[*i*]>＝*x*，则我们可以用 *f*[*i*]－*x* 的答案＋1 和 *x*－*f*[*i*－1]的答案＋1 来递归出我们现在的答案。

2797: [Poi2012]Squarks

我们设给定的和为 $a_1,a_2\cdots\cdots$，且 $a_i<=a_{i+1}$。设未知数分别为 $x_1,x_2\cdots\cdots x_n$。我们不难发现 $a_1=x_1+x_2$，$a_2=x_1+x_3$。那么在 $a_k(k>2)$中必存在 $a_i=x_2+x_3$。我们可以枚举这个 *i* 的位置，进而我们可以求出 x_1，有了 x_1 以后，只需要用堆维护剩余的数即可依次求出 x_2 到 x_n。然后判重输出就可以了。

2799: [Poi2012]Salaries

我们首先根据读入的数据，确定根节点，根节点上的数一定是 *n*。下面我们考虑按照递增的顺序扫描 1 到 *n* 的每个数能否被确定。开一个栈记录当前的数，对于一个数 *i*：

①.这个数当前还没有被确定，我们则把这个数加入堆栈中；

②.这个数已经确定，我们找到它所在的结点记为 *p*。考虑 *p* 的每个儿子，定义未计算结点为一个上面的数还没有被确定的节点。*s* 为 *p* 的所有未计算节点的子树大小之和，如果 *s*＝0 则忽略下面的所有过程。定义一个变量 *bad* 表示有 *bad* 个数，一定出现在之前计算的节点中，且这 *bad* 个数的具体位置不能确定。如果堆栈中的数的个数加上 *bad* 刚好等于 *s*，那么我们进行下面的计算：从 *p* 出发，沿着它的儿子走，如果当前 *p* 的未计算儿子为 1 个，则把这个儿子的数值赋值成栈顶元素，然后栈顶元素退栈，*s* 减 1，并向下重复这样的过程，否则退出。当结束这个过程时，*bad* 加上栈中元素的个数，*bad* 减去此时的 *s*，清空堆栈。

整个做法时间复杂度为 *O*(*n*)。

2800: [Poi2012]Leveling Ground

首先差分原序列，定义 $s[i]=a[i]-a[i-1]$，$s[n+1]=-a[n]$。我们发现把$[L,R]$都加 A，实际就是把 $s[L]$加 A，把 $s[R+1]$减 A，我们最终的目标是使得 $s[i]=0$，$1<=i<=n+1$。
对于无解的情况：存在一个 $s[i]$使得 $s[i]$无法整除 $\gcd(A,B)$。

对于有解的情况：我们设在位置 i 上，A 使用了 $x[i]$次，B 使用了 $y[i]$次。则答案就是 $\frac{\sum_{i=1}^{n+1}(|x[i]|+|y[i]|)}{2}$。我们先考虑依次满足每个 $s[i]$，则有 $Ax[i]+By[i]+s[i]=0$，这个先用扩展欧几里得随便解出一组解，比如 $x[i]=x$，$y[i]=y$，进而我们要求出一组最小的 $|x[i]|+|y[i]|$，考虑已求得的一组解，则 $x+kB$，$y-kA$ 都是满足条件的解。我们可以发现当 k 从负无穷向正无穷移动的过程中，$z=|x|+|y|$ 是一个单峰函数，因此我们可以通过三分的方式求出最小的 z，及对应的 $x[i]$，$y[i]$。

求出 $x[i]$和 $y[i]$以后我们已经满足了对应任意的 $s[i]$有 $s[i]=0$，但是由于 A 或者 B 操作每次都是操作两个数，一正一负，所以我们还需要满足 $\sum_{i=1}^{n+1}x[i]=\sum_{i=1}^{n+1}y[i]=0$。我们不妨假设现在 $\sum_{i=1}^{n+1}x[i]>\sum_{i=1}^{n+1}y[i]$，易证 $\sum_{i=1}^{n+1}x[i]$ 是 B 的倍数，$\sum_{i=1}^{n+1}y[i]$ 是 A 的倍数，考虑一个二元组$(x[i],y[i])$，我们可以把 $x[i]$减上 B，把 $y[i]$加去 A，这样也是合法的。因此我们通过这种调整的方式进行调整，直到 $\sum_{i=1}^{n+1}x[i]=\sum_{i=1}^{n+1}y[i]=0$，调整的方式是每次选择一个调整以后对答案影响最小的二元组进行调整，对于最小值的选择，我们可以把所有的二元组都扔进一个堆就可以了。

这个算法有一个问题就是我们很难保证调整的次数不会特别大，但是没关系，使用此方法可以通过所有的测试数据。

2801: [Poi2012]Minimalist Security

容易观察到不同连通块之间互不影响，因此我们考虑处理每个连通块再把答案相加即可。对于一个连通块如果我们选定一个起始点并把它的点权修改为 0，那么剩下点的点权都可以唯一确定。我们给每个点染一个颜色，其中起始点是黑色，与起始点相邻的点都是白色，然后从这些白色的点出发，与白色相邻的点再染成黑色，以此类推，进行一次宽搜即可。我们发现由于起始点的点权只能变大，这就导致了所有黑色点的点权都会一起变大，白色点的点权都会一起变小，我们只需要根据题目的条件确定每个点的变化范围，然后把所有点的变化范围取个交集即可，如果交集是空的那么显然无解。当然还有一些需要特判的情况，由于每个连通分量不一定是一棵树，因此可能会存在这样的一条边，它的两个端点的颜色是相同的，那么这个时候这两个点的变化就被唯一确定成了一个值，而不是一个范围，所以要把整个范围与这个值取交集，再去进行下面的判断。当然这还不够，如果一条边连接的两个点颜色不同，那么如果当前的点权加起来不等于边权的话也是无解的，因为无论怎样变都是一加一减，总和不变，如果一条边连接的两个点颜色相同，并且当前两个点点权之和的奇偶性和边权的奇偶性不同的那么也是无解的。时间复杂度 $O(n+m)$。

2802: [Poi2012]Warehouse Store

贪心就可以了，记两个变量 A 和 B，表示扫描过的库存的总和以及使用的总和，如果当前的这个客户的需求加上 B 不大于 A，则满足他，否则如果满足的用户中需求最大的大于当前用户的需求则把这个用户的需求替换最大的那个用户，用堆维护就行。

2803: [Poi2012]Prefixuffix

我们发现一对字符串是循环同构的，它们一定长成 AB 和 BA 这个样子，那么我们考虑枚举这个 B 串的起始位置，我们将得到一个 B 串的长度记为 len，有一个事实就是对于当前位置 i 得到的 len，那么 $i-1$ 位置 len'一定不会大于 $len+2$，证明也很简单，首先 $len+2$ 肯定是可以取到的，比如我们枚举到 i 这个位置的时候：

$abcd|dabcd$，这时候的 $len=3$，那个竖线表示整个字符串的对称中心，显然是取 abc 这个串，刚好前后各一个。

我们在 $i-1$ 位置又遇到了一个 d，那么我们如果得到了：

$dabcd|dabcd$，这个时候我们就可以选择 $dabcd$，$len'=5$，下面我们证明 len 不可能被加 3 或者被加更多。

假如有一个字符串 B 后面紧跟四个字符 $abcd$（这都是变量，不是真的字母）。那么我们目前得到的答案就是：$Bab|cdBe$，假如我们在前面遇到了一个字符 f，则字符串变成：

$fBab|cdBe$，如果 len 要增加 3，则有 $fBab=cdBe$，也就是 $Bab=dBe$，那么显然这个 B 不是上一次得到的最长的 B，因为可以把 B 的右面加上一个 a，前面加上一个 d 得到一个更长的 B'，这不符合我们算法的定义。
由于加 3 不可行，所以比 3 大的就都不可行了。

因此我们只需要从上一次得到的 len，把 $len+2$，然后得到一个可行解，过程是如果 $len+2$ 不可行，则枚举 $len+1$，一直到 0 为止。算出这个以后如果在 i 这个位置左面的字符串（也就是 $S[1,i-1]$）能和对称过去右面的匹配的话，更新答案即可，判两个字符串是否相等用 HASH 就可以了，选个好点的种子和模数就没问题。时间复杂度 $O(n)$。

2806: [Ctsc2012]Cheat

把所有的文章连在一起，中间用一个既不是 0 也不是 1 的字符隔开，然后把整个大串扔进后缀自动机。对于一个等待检验的字符串，我们设第 i 个字符能最多能往前匹配 $d[i]$个字符，这个 $d[i]$可以在后缀自动机上很容易求出。我们发现如果一个 L 可行，那么 $L-1$ 一定可行，因此想到二分答案，考虑当前的 L 如何检验。定义状态 $dp[i]$表示前 i 个字符最多能匹配多少个，则有 $dp[i]=\max(dp[i-1],dp[j]+i-j)$，$(i-d[i]<=j<=i-L)$。这样做时间复杂度是 $O(n^2)$的，由于本题的性质不难发现 $d[i]-d[i-1]<=1$，因此对于 $i<j$，$i-d[i]$一定小于等于 $j-d[j]$，所以可以对上面那个方程进行整理，则有 $dp[i]=\max(dp[i-1],(dp[j]-j)+i)$。其中 $dp[i-1]$我们可以 $O(1)$转移得到，后面的那个用单调队列也可以做到均摊 $O(1)$，因此时间复杂度为 $O(n+\Sigma m\log m)$，其中 n 为文章总长，m 每次的询问长度。

2815: [ZJOI2012]灾难

我们构造一种叫做灾难树的东西，其中以 i 为根的子树的所有节点表示如果 i 死了，那么以 i 为根的子树中所有的节点也会死。我们根据输入数据，不妨为所有生产者都建立一个共同的食物来源，太阳。按照输入数据，如果 a 吃 b，则连边 a 到 b。太阳连所有生产者，太阳为根节点。我们对原图进行拓扑排序，然后建立一棵灾难树。我们按照拓扑序进行处理，设当前处理节点为 p，我们找到所有 p 的所有食物节点，也就是原图的反向边，我们设这些点的最近公共祖先为 q，那么显然当 q 死了的时候，p 才会死，所以我们在灾难树中连边 q 到 p。最后，我们整理这个灾难树，求出每个节点的子树大小，不难发现子树大小减一就是对应节点的答案。

2816: [ZJOI2012]网络

没什么技术含量的破题，把每种颜色分开考虑，对每种颜色建一棵 LCT 就行了。

2818: Gcd

我们设第 i 个素数为 $prm[i]$，第 i 个欧拉函数值为 $phi[i]$，$sphi[i]$为前 i 个欧拉函数的和。共有 m 个质数，显然 $ans=\Sigma sphi[n/prm[i]]\times 2-1$，$1\leqslant i\leqslant m$

2819: Nim

水题，用树链剖分或者是 LCT 维护一下异或和就可以了。

2820: YY 的 GCD

我们首先考虑如何求 $\sum_{i=1}^{n}\sum_{j=1}^{m}1\left(\gcd(i,j)=d\right)$，不妨设 $n\le m$。

$$\sum_{i=1}^{n}\sum_{j=1}^{m}1\left(\gcd(i,j)=d\right)$$

$$=\sum_{i=1}^{\left\lfloor\frac{n}{d}\right\rfloor}\sum_{j=1}^{\left\lfloor\frac{m}{d}\right\rfloor}1\left(\gcd(i,j)=1\right)$$

$$=\sum_{i=1}^{\left\lfloor\frac{n}{d}\right\rfloor}\sum_{j=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\sum_{x|\gcd(i,j)}m(x)$$

$$=\sum_{i=1}^{\left\lfloor\frac{n}{d}\right\rfloor}\sum_{j=1}^{\left\lfloor\frac{m}{d}\right\rfloor}\sum_{x|i\text{且}x|j}m(x)$$

$$=\sum m(x)\sum_{1\le i\le\left\lfloor\frac{n}{d}\right\rfloor\text{且}x|i}\sum_{1\le j\le\left\lfloor\frac{m}{d}\right\rfloor\text{且}x|j}1$$

$$=\sum_{i=1}^{\left\lfloor\frac{n}{d}\right\rfloor}m(i)\left\lfloor\frac{n}{i}\right\rfloor\left\lfloor\frac{m}{i}\right\rfloor。$$

接下来我们考虑如何求 $\sum_{i=1}^{n}\sum_{j=1}^{m}1\left(\gcd(i,j)\text{为素数}\right)$。

$$\sum_{i=1}^{n}\sum_{j=1}^{m}1\left(\gcd(i,j)\text{为素数}\right)$$

$$=\sum_{d\text{为素数}}\sum_{i=1}^{n}m(i)\left\lfloor\frac{n}{di}\right\rfloor\left\lfloor\frac{m}{di}\right\rfloor$$

我们考虑每个分母上的 di 对答案的贡献，则原式

$$=\sum_{i=1}^{n}\left\lfloor\frac{n}{i}\right\rfloor\left\lfloor\frac{m}{i}\right\rfloor\sum_{d|i}m\left(\frac{i}{d}\right)$$

我们设 $g(i)=\sum_{d|i}m\left(\frac{i}{d}\right)$，则原式

$$=\sum_{i=1}^{n} g(i)\left\lfloor\frac{n}{i}\right\rfloor\left\lfloor\frac{m}{i}\right\rfloor$$

我们考虑如何预处理 $g(i)$，设 d 为素数，易证：

当 $d \mid i$ 时 $g(di)=m(i)$，

否则 $g(di)=m(i)-g(i)$。

这正好对应筛法的两种情况，因此 $g(i)$ 可以在求 $m(i)$ 时顺便求出。

我们知道 $\left\lfloor\frac{n}{i}\right\rfloor$ 的不同的取值个数是在 $O(\sqrt{n})$ 数量级的，因此我们可以把 $\left\lfloor\frac{n}{i}\right\rfloor\left\lfloor\frac{m}{i}\right\rfloor$ 进行分段处理，再预处理 $g(i)$ 的前缀和即可。记 p 为 n，m 中出现的最大数，数据组数为 T，时间复杂度为 $O\left(p+T\left(\sqrt{n}+\sqrt{m}\right)\right)$。

2827：千山鸟飞绝

首先把坐标离散成整数，然后离线所有操作。把每次移动看成一个删除操作和一个插入操作，之前给出的每只鸟看成 n 个插入操作，最后加入 n 个删除操作。每个操作记录运动的位置，是哪只鸟，移动的时间，是插入还是删除。把所有操作按照运动位置排序，对于运动位置一样的放在一起处理，记每只鸟的插入时间为 L，删除时间为 R 用[L,R)的信息来更新答案即可，用线段树维护每个时刻有多少只鸟，最大的威武值以及次大的威武值，用堆维护当前存在的威武值。

2849：和谐串

这个题规律很容易找的，打表前三四项就能看出来。我们设 $f[i]$表示当 $n=i$ 时的答案，则有 $f[1]=\frac{2}{3}$，$f[i]=f[i-1]\times\frac{2i}{2i-1}$。也就是说 $f[i]=\frac{2n!!}{(2n+1)!!}$，两个!是双阶乘。有一个神公式是 $\frac{2n!!}{(2n-1)!!}\approx\sqrt{pn}$，那么 $f[i]\approx\sqrt{\frac{p}{4n}}$，这个东西的精度误差是 $\frac{1}{n}$，由于题目限制的精度误差是 10^{-6}，所以对于 n 小于等于 10^7 的暴力算，剩下的用公式算就行了。

2851：极限满月

我们考虑将 n 个数建成一棵树，设立第 0 号元素。题目中集合 B_i 是一些集合的并集，我们不妨设这些集合为 $B_{a1},B_{a2},\cdots\cdots,B_{ak}$，那么我们只需要求出 $B_{a1},B_{a2},\cdots\cdots,B_{ak}$ 的 LCA，设这个公共祖先为 j，那么我们在树中加入一条边，这个边的两个顶点分别为 i 和 j。做完这些以后我们考虑一个询问，我们发现对于一个询问，实际上是求一些树链的并，我们发现直接求是很困难的，因此我们考虑求原树的 DFS 序列。我们把这个询问中的所有点按照它们在 DFS 序中的位置重新排序，对于第 i 和第 $i+1$ 个点，我们设 $j=i+1$，那么第 j 个点对答案是贡献就是 j 到根节点的距离减去 i 和 j 的 LCA 到根节点的距离，第 1 个点的贡献就是它自身到根节点的距离。

2870：最长道路 tree

树分治就可以了，分过根节点和在子树两种情况。时间复杂度 $O(n\log n)$。

2875：[Noi2012]随机数生成器

矩阵乘法不多说了，中间会涉及两个 64 位的整数相乘，把其中一个用二进制展开就可以了。

2876：[Noi2012]骑行川藏

我们考虑分配一微小部分的能量 ΔE 可以使某一段路的行驶时间减少多少，设风速为 v_0 则有：

初始能量：$E_0 = ks(v - v_0)^2$

增加后的能量：$E = ks(v + \Delta v - v_0)^2$

能量的改变：$\Delta E = E - E_0$

联立以上三式有：$\Delta E = ks\Delta v(2v - 2v_0 + \Delta v) = 2ks\Delta v(v - v_0)$

整理得：$\Delta v = \dfrac{1}{2ks(v - v_0)}\Delta E$

接下来我们考虑时间的变化：

初始时间：$t_0 = \dfrac{s}{v}$

增加能量后的时间：$t = \dfrac{s}{v + \Delta v}$

时间的改变：$\Delta t = t - t_0$

联立以上三式有：$\Delta t = \dfrac{s\Delta v}{v(v + \Delta v)} = \dfrac{s\Delta v}{v^2}$

将上式带入求得的 Δv 有：$\Delta t = \dfrac{1}{2kv^2(v - v_0)}\Delta E$

我们不难发现当初速度为 0 时，Δt 为无穷大，所有路段的增量相同且都为无穷大。所以我们猜测在按照最优方案进行设计使每段路的每个微小的 ΔE 带来的收益，也就是使行驶时间的减少量，这个减少量都相等。因此对于每段路都有 $\dfrac{1}{2kv^2(v - v_0)}$ 为定值，我们可以考虑二分这个定值的分母，因为随着定值的分母的增大，速度在增加，时间在减少，能量在增加，这些都是单调的。考虑当前二分等待检验的值 T 是否可行，对于其中一段路有 $2kv^2(v - v_0) = T$，则 $v^2(v - v_0) = \dfrac{T}{2k}$，我们设 $f(v) = v^2(v - v_0)$，则 $f'(v) = v(3v - 2v_0)$，令 $f'(v) = 0$ 有：$v_1 = 0$，$v_2 = \dfrac{2}{3}v_0$。因此我们可以得到我们所需要的方程的根一定大于 v_0，所以我们可以二分这个方程的根。求出了 v 以后，我们把它带回到题目中的式子里也就求出了这段路所消耗的能量。将所有消耗的能量加起来如果小于等于题目给定的总能量则当前二分的分母可行，否则不可行。我们记 m 为分母的二分次数，p 为解方程的二分次数，则时间复杂度为 $O(nmp)$，取 $m = p = 100$ 即可达到精度，通过所有测试数据。

2878：[Noi2012]迷失游乐园

记 $up[i]$表示向上走的期望长度，$down[i]$表示向下走的期望长度，$fa[i]$表示父亲，$son[i]$表示儿子集合，$chn[i]$表示儿子个数，$d[i]$表示度数。首先考虑无环的情况，有：

$$down[i]=\frac{\sum_{j\in son[i]}\left(dist[i][j]+down[j]\right)}{chn[i]}$$

$$fa[i]=j$$

$$up[i]=dist[i][j]+\frac{up[j]\times\left(d[j]-chn[j]\right)+down[j]\times chn[j]-dist[i][j]-down[i]}{d[j]-1}$$

$$ans=\frac{1}{n}\sum_{i=1}^{n}\frac{up[i]\times\left(d[i]-chn[i]\right)+down[i]\times chn[i]}{d[i]}$$

对于有环的情况，我们先找到环，记环上的点依次为 $c[1]$，$c[2]$，……，$c[m]$。我们枚举每个环上的点，求出以 $c[i]$为根的子树每个节点的 $down[i]$，接下来我们只需要求出环上点的 $up[i]$，然后再用上面的方法计算答案即可。我们考虑如何计算环上每个点的 $up[i]$，我们可以从点 i 沿着顺时针方向走到某个点，然后从这个点为根的子树下去，或者是这个点没有子树，那么继续向前走。维护一个 p 值表示走到当前点的概率是多少，最后把顺时针逆时针分别计算一次即可。时间复杂度为 $O(n+m^2)$，其中 m 为环上节点的个数。

2879：[Noi2012]美食节

我们考虑每个厨师做菜的顺序，这个厨师按照顺序每道菜菜所需时间为 d_1，d_2，d_3，……，d_n。则对于第一个人的等待时间为 d_1，第二个人为 d_1+d_2，第三个人为 $d_1+d_2+d_3$……所以 d_1 对整个时间的贡献度为 $d_1\times n$，也就是第 i 道菜对总时间的贡献为$(n-i+1)\times d_i$。建立附加源点 S，附加汇点 T。把每个厨师拆成 p 份（p 为每道菜需求量的总和），其中第 i 份表示对时间的贡献为 i 倍。S 向每道菜连容量为所需数量，费用为 0 的边。第 j 个厨师的第 k 倍点向 T 连容量为 1，费用为 0 的边。第 i 道菜向第 j 个厨师的第 k 倍点连容量为 1，费用为 $cost[i][j]\times k$ 的边，求出最小费用流就是答案。但由于点数过多会导致超时。我们可以这样优化，考虑到 SPFA 的增广过程是按照费用从小到大进行的，也就是说如果 1 倍点还没有使用的话，那么 2 倍点一定不会使用。所以我们可以先只建立 1 倍点的边，当 1 倍点增广过后，再添加 2 倍点的边，以此类推即可。这样做可以通过所有测试数据。

2882：工艺

把原串在后面接一遍，现在长度就翻倍了，然后跑一遍后缀自动机，从根节点出发沿着最小的数往下走 n 个，得到那个节点所在的编号，然后继续输出 n 个数就可以了。

2896：桥

由于题目没有要求在线算法，因此我们可以考虑离线所有操作。我们设删掉所有操作中要求删掉的边得到图 G，我们将 G 收缩双连通分量。然后考虑询问，查询操作还是查询，删边变成了插边。这样我们需要一个数据结构支持：

①. 询问两个点之间有多少条边。

②. 将两点之间所有边权赋值为 0。

我们把边权拆成点权，然后用 LCT 做就可以了。

2956：模积和

首先不考虑 $i\neq j$ 这个条件，我们记 $f(n,m)=\sum_{i=1}^{n}\sum_{j=1}^{m} n \bmod i \times m \bmod j$，则：

$$f(n,m)=\sum_{i=1}^{n}\sum_{j=1}^{m}\left(n-i\left\lfloor\frac{n}{i}\right\rfloor\right)\times\left(m-j\left\lfloor\frac{m}{j}\right\rfloor\right),$$

把括号展开得：

$$f(n,m)=n^2m^2-m\sum_{i=1}^{n} i\left\lfloor\frac{n}{i}\right\rfloor-n\sum_{i=1}^{m} i\left\lfloor\frac{m}{i}\right\rfloor+\sum_{i=1}^{n}\sum_{j=1}^{m} ij\left\lfloor\frac{n}{i}\right\rfloor\left\lfloor\frac{m}{j}\right\rfloor。$$

上面这个显然可以在 $O\left(\sqrt{n}+\sqrt{m}\right)$ 的复杂度下解决。

不妨设 $n\leqslant m$，我们考虑多算的，需要减去的一部分：

$$g(n,m)=\sum_{i=1}^{n}\left(n-i\left\lfloor\frac{n}{i}\right\rfloor\right)\times\left(m-i\left\lfloor\frac{m}{i}\right\rfloor\right),$$

这个可以用跟上面一样的方法计算，最后的答案就是 $f(n,m)-g(n,m)$。

2969：矩形粉刷

我们设随机粉刷一次之后，格子(i,j)被着色的概率为$E(i,j)$，则染k次之后格子(i,j)被着色的概率就为$D(i,j)=1-(1-E(i,j))^k$。而被染色格子的期望个数的和就为所有$D(i,j)$的和，因此我们只需要求出$E(i,j)$即可解决此题。我们首先考虑一维情况。如果是一行n个格子，随机染一次染到第i个格子的概率为多大。我们设随机染色的区间的两个端点为x和y，显然x和y的取法共有n^2种。当$x\leqslant y$时，有$i\times(n-i+1)$种，当$x\geqslant y$时有$i\times(n-i+1)$种，以上两种情况多算了$x=y=i$时的一种。所以一行n个格子，第i个格子被染的概率就为$(2\times i\times(n-i+1)-1)/n^2$。然后把这个直接扩展到二维则$E(i,j)=(2\times i\times(n-i+1)-1)\times(2\times j\times(m-j+1)-1)/n^2m^2$。

2982：combination

记$p=10007$，原式等于$\frac{n!}{m!\times(n-m)!}\bmod p$。我们考虑分离出所有的$p$，也就是求三个阶乘共包含有多少个$p$，分母算正的，分子算负的，其中$p\times p$这种要算两个，如果最后$p$的个数大于等于1个直接输出0，否则记此时有$k$个$p$，我们去算一下$p^{-k}$的逆元。由于三个阶乘等价，考虑如何求出$n!$里去掉所有的$p$以后的值，也就是求

$f(n)=\prod_{i=1}^{n} j\left(i=j\times p^k, j\text{与}p\text{互质}\right)$。记$t(i)=\begin{cases} i(i \bmod p\neq 0) \\ 1(i \bmod p=0)\end{cases}$，$T(n)=\prod_{i=1}^{n} t(i)$，则有$f(n)=\begin{cases} T(n)\times f\left(\left\lfloor\frac{n}{p}\right\rfloor\right) & (n\geq p) \\ T(n) & (n<p)\end{cases}$。

3011：[Usaco2012 Dec]Running Away From the Barn

我们考虑 dfs 整棵树，我们用一个栈记录当前经过的点。我们可以发现栈中的点就是当前等待处理的节点到根节点的路径，我们不妨设整个栈为[1,*top*]，我们二分求得一个最小的 p，使得[*p*,*top*]这些点到达当前节点的距离都不超过题目中的限制，然后我们对[*p*,*top*]对应的点的所有答案都加 1，这个过程可以用一个线段树维护，退栈的时候查询退栈节点的最终答案，并且把线段树中的值清零即可。

3012: [Usaco2012 Dec]First!

我们考虑把所有的字符建到一个字典树中，考虑字符串 i 和字符串 j，我们发现如果 i 是 j 的前缀那么 j 不可能成为字典序最小的字符串。接下来我们逐一考虑每个有可能成为字典序最小的字符串。我们找到这个字符串在字典序中从它到根节点的路径，接下来我们只需要看每个节点的儿子情况，这个字符串在那个位置的字符为 x，当前节点有一个字符为 y，其中($x \neq y$)，我们考虑如果 i 要成为字典序最小的字符串，那么 x 就必须排在 y 的前面，于是我们连一条由 x 到 y 的边。那么处理完这些以后我们就得到了一个包含 26 个节点的有向图，如果这个有向图可以拓扑排序，那么这个字符串就可以成为字典序最小的字符串。至于最开始处理前缀的包含关系，我们可以先把所有的字符串按照字典序进行排序，那么如果 i 是 j 的前缀 i 就一定排在 j 的前面了。

3013: [Usaco2012 Nov]Balanced Cow Breeds

跟 3017 是一个题。

3016: [Usaco2012 Nov]Clumsy Cows

我们发现去掉匹配的括号，剩下的括号一定是))))))((((这种形式。我们记多余的右括号有 x 个，多余的左括号有 y 个，那么答案就是 $\left\lfloor \frac{x+1}{2} \right\rfloor + \left\lfloor \frac{y+1}{2} \right\rfloor$。

3017: [Usaco2012 Nov]Cow Breeds

我们观察不难发现，无论怎么标号，当前 H 和 G 所选的括号都必须是左括号个数多余右括号个数，且我们把左括号看成 1，右括号看成−1，对于任意的前缀和都必须非负，这样才能保证合法。然后就可以用 DP 解决，定义状态 $dp[i][j]$表示前 i 个括号中，其中 H 选的左括号比右括号多 j 个，那么 G 的情况也被唯一确定，我们只需要根据第 i 个括号是左括号还是右括号，从 $dp[i-1][j-1]$，$dp[i-1][j+1]$，$dp[i-1][j]$进行转移即可，最终答案就是 $dp[n][0]$。

3018: [Usaco2012 Nov]Distant Pastures

直接求最短路就行了，没啥说的。

3038: 上帝造题的七分钟 2

观察到每个数越来越小最终变成 1，而且一个数从很大被开平方变成 1 不会经过太多次，所以求和操作可以用一个树状数组维护，开平方只要暴力开就可以，如果第 i 个数变成了 1，就把 i 和 $i+1$ 捆在一起，这个过程用并查集维护就可以了。

3042: Acting Cute

跟 1737 是一个题。

3043: IncDec Sequence

设原序列为 $a[i]$，定义差分序列 $b[i]=a[i]-a[i-1]$。我们发现我们的目标是使得 $b[2..n]$都为 0，我们记差分序列中所有 $b[i]>0$ 的和为 A，所有 $b[i]<0$ 的和的相反数为 B，则第一问答案就是 $\max(A,B)$，第二问答案就是 $\max(A,B)-\min(A,B)+1$。

3048: [Usaco2013 Jan]Cow Lineup

我们考虑两个位置 A 和 B，不妨假设 $A<B$，我们设有这样的一个 a，将$[a,A]$不等于第 A 个数的数都删掉以后，删掉的总个数不超过 k 个，a 为最小的满足条件的值。同理我们求出 B 对应的 b。那么当 $A<B$ 时有 a 不大于 b，证明如下。如果$[A,B]$内所有的数都相等那么显然 $a=b$，否则对于 B 来讲，第 A 个位置所对应的数一定要被删掉，所以 b 必须大于 a，综上 a 不大于 b。有了这个发现以后，我们只需要从头到尾扫描整个数组，随着 i 的增加，能满足条件的最小左端点一定在单调地右移，通过对原数组离散化或者直接使用 STL 中的 *map* 可以使整个算法的时间复杂度为 $O(n\log n)$。

3049: [Usaco2013 Jan]Island Travels

预处理任意两个连通块之间的最短路，然后状压 DP 求解。

3050: [Usaco2013 Jan]Seating

裸题，直接线段树维护区间最长连续 1 的个数即可。

3060: [Poi2012]Tour de Byteotia

我们考虑所有两个点编号都大于 k 的边，把这样的边取出来，建新图，然后收缩双连通分量，对每个点重新进行标号。最后我们再把所有的边拿出来，如果一条边连接的两个点新标号相同则忽略这样的边，如果不属于一个连通块则连上这两条边，否则答案加 1，维护连通性用并查集即可。

3061: [Usaco2013 Feb]Partitioning the Farm

考虑到 n 比较小，我们可以暴力枚举第 i 行是否进行切割，然后对于剩下的木条在列里用经典的贪心二分做大即可，时间复杂度 $O(2^n n\log \boldsymbol{V})$，其中 $\boldsymbol{V}$ 表示所有方格数的和。

3062: [Usaco2013 Feb]Taxi

行走路线分为两种，第一种是车上有牛的，第二种是无牛的。第一种可以根据输入数据直接算出，第二种只需要把所有区间排序以后贪心计算即可。

3063: [Usaco2013]Route Designing

考虑到一个合法的行走路线，左侧的点标号严格递增，右侧的点标号严格递增，因此可以把每条边按照左边编号为第一关键字，右边编号为第二关键字排序，然后用每条边更新路线即可。

3067: Hyperdrome

我们发现一个子串可以重组成回文串的充要条件是每种字母都出现偶数次或者只有一种字母出现奇数次。我们记 $a[i]$为一个 52 位的二进制数，表示字符串的第 1 到第 i 位的每个字符出现的奇偶情况，如果一个字符出现了奇数次则它对应的那位就是 1 否则那位是 0。这样我们就可以在 $O(1)$的时间判断一个串是否合法，$[l,r]$合法的充要条件是 $a[r]\ xor\ a[l-1]$等于 0 或者等于 2 的某次幂。我们首先考虑 $a[r]\ xor\ a[l-1]=0$ 的情况，这种情况我们只需要把 $a[i](0<=i<=n)$按照从小到大进行排序，然后通过一次线性扫描即可解决。接下来我们考虑 $a[r]\ xor\ a[l-1]$（此时 a 数组已经按照从小到大的顺序排序了）的二进制只有一位是 1 的情况。我们不妨逐位进行枚举，假设要求第 T 位为 1，则 $a[r]\ xor\ a[l-1]=2^T$。我们设 $b[i]=a[i]\ xor\ 2^T$。这样 $a[r]\ xor\ a[l-1]=2^T$ 就变成了 $a[r]\ xor\ b[l-1]=0$。我们发现只要把 b 数组也排序，然后就仍然可以通过一次线性扫描得到答案，这样我们就有了 $O(n\log n+52n\log n)$时间复杂度的做法，但是无法通过所有测试数据。我们发现 b 数组没有必要进行整体排序，对于一个 $b[i]=a[i]\ xor\ 2^T$，$b[i]$要么是 $a[i]$减去 2^T 要么是 $a[i]$加上 2^T，因此我们可以把减去的分成一组，把加上的分成一组，这样这两组中的任意一组都是有序的，因此我们可以在 $O(n)$的时间内归并两个数组，这样我们就把一轮排序的时间降低了，优化以后的时间复杂度变为 $O(n\log n+52n)$，可以通过所有测试数据。

3074: [Usaco2013 Mar]The Cow Run

我们以 0 为原点，将位置分为正负两个部分，并将两部分分别排序，记左面有 n 个点，右面有 m 个点。我们根据题目给定的计算规则可以发现对于第 i 次行走的距离 d，对答案的贡献为 $d\times(n+m-i)$。考虑用动态规划解决后面的问题，$dp[i][j][0]$表示左面从小到大的 i 个点右面从小到大的 j 个点已经走完，且当前在左面的第 i 个点，最小花费值，类似定义 $dp[i][j][1]$。边界条件是 $dp[0][0][0]=dp[0][0][1]=0$，我们每次可以用 $dp[i][j]$来更新 $dp[i+1][j]$和 $dp[i][j+1]$，最后在 $dp[n][m][0]$和 $dp[n][m][1]$中选一个较小的最为答案即可。时间复杂度 $O(n^2)$。

3075: [Usaco2013]Necklace

考虑将小串用 kmp 或者 AC 自动机进行处理，处理出失败指针。定义状态 $dp[i][j]$表示大串的第 i 个字母必须保留，且目前处于状态 j 能保留的最多字符。边界条件 $dp[0][root]=0$，每次可以用 $dp[i][j]+1$ 来更新 $dp[i'][j']$，这个只需要枚举下一个字母是什么即可 j 到 j'的转移可以预处理一下。

3076: [Usaco2013]Hill Walk

我们考虑用扫描线维护每条线段，把每条线段拆成一个插入事件和一个删除事件，我们发现只要求出了从每条线段掉下去掉到哪里便可解决此题。这个可以在扫描线的时候把还没有删除的线段扔进一个平衡树维护，然后通过查找前驱的方式实现。

3091：城市旅行

我们首先将问题一般化，考虑在一个序列[1,n]上进行上面的操作，这样原来的每个结点就变成了一个整数，记第 i 个整数为 a_i，$Sum(l,r)=\sum_{i=l}^{r}a_i$，则[1,$n$]的答案就为$T=\frac{\sum Sum(l,r)}{\sum 1}(1\le l\le r\le n)$。我们考虑每个 a_i 对答案的贡献，易知只有当 $l\le i\le r$，a_i 才会被计算到分子中，因此每个 a_i 计算了 $i\times(n-i+1)$ 次，所以 $T=\frac{\sum_{i=1}^{n}a_i\times i\times(n-i+1)}{C_{n+1}^{2}}$。仔细观察可以发现，这个问题可以用线段树解决，而用线段树解决的前提是根节点的信息可以在极短的时间内由两棵子树合并而来，我们首先考虑叶子节点，对于叶子节点 i，显然有$T=a_i$。接下来我们考虑一个区间[1,n]，设它由两个区间 A 和 B 合并而来，对于每个区间记：

$size=n$，

$sum=\sum_{i=1}^{n}a_i$，

$sumL=\sum_{i=1}^{n}i\times a_i$，

$sumR=\sum_{i=1}^{n}(n-i+1)\times a_i$，

ans 为整个区间的答案，也就是上面我们说的 T，对于两个区间合并后的大区间有：

$size=size_A+size_B$，

$sum=sum_A+sum_B$，

$sumL=sumL_A+sumL_B+size_A\times sum_B$，

$sumR=sumR_A+sumR_B+size_B\times sum_A$，

$ans=ans_A+ans_B+sumL_A\times size_B+sumR_B\times size_A$。

推导过程很容易，这里就不写了，都是简单的代数运算。这样如果原问题不是一棵树，而是一个数列的话，我们就有了预处理时间复杂度$O(n)$，单次询问时间复杂度$O(\log n)$的算法。

接下来我们考虑修改操作，对区间[1,n]加每个 a_i 都加 d，则：

$sum=sum+d\times size$，

$sumL=sumL+d\times\frac{size\times(size+1)}{2}$，

$sumR=sumR+d\times\frac{size\times(size+1)}{2}$，

$ans=ans+d\times\frac{size\times(size+1)\times(size+2)}{6}$。

这些的推导过程依然非常简单，这里就不详细说明了。到此为止我们就有了在一个序列上，预处理时间复杂度$O(n)$，单次询问时间复杂度$O(\log n)$，单次修改$O(\log n)$复杂度的算法。

接下来我们考虑如何把序列扩展到树上，我们发现使用 LCT 可以很轻松完成上面的所有操作，只要用与线段树一样的方法在 LCT 中的伸展树中维护上面提到的信息即可，而删边，加边和询问都是 LCT 的基础操作，这里就不多说了。这样这道题就在 $O\left((n+m)\log n\right)$ 的时间复杂度下得到解决。

3108：[cqoi2013]图的逆变换

我们发现新图中的每条边对应原图中的两个顶点，对于新图的一条边连接的两个顶点 A 和 B，在原图中一定是 (a,b)，(b,c) 这两条边，也就是第一条边的终点和第二条边的起点相同。我们把每条边拆成两个点，对于题目输入的边，使用并查集合并第一条边的终点和第二条边的起点，然后用 $O(n^2)$ 的时间扫描两个顶点 i 和 j，如果如果 i 的终点和 j 的起点在一个集合中，但是题目并没有给 i 和 j 之间连边，则这个图不合法，剩余的情况都合法，注意特判输入的边两个顶点相同的情况，这需要将这两个顶点拆成的四个点都合并起来。

3109：[cqoi2013]新数独

我们观察到每个 3×3 的小方格内都有大量的限制，不妨先考虑预处理每个 3×3 小方格内的合法状态，可以发现被 12 个不等关系限制以后的合法状态并不会特别多。然后就变成在 9×9 的方格上，放 9 个 3×3 的小方格，这个继续在合法的状态内爆搜就行了，注意任意时刻如果有某一行或者某一列的数出现矛盾则剪枝。

3110：[Zjoi2013]K 大数查询

考虑建立二维线段树，外层线段树记录数的范围，内层线段树记录位置，这样我们发现把 $[L,R]$ 都插入一个 d，实际上就是把外层线段树包含 d 的节点进行更新，易知这样的节点只有 $\log n$ 个，对于更新过程，实际是把每个节点的内层线段树的 $[L,R]$ 的值都加 1，易知内层节点每次只会更新 $\log n$ 个，因此修改的复杂度为 $O(\log^2 n)$，询问与这同理，复杂度相同。但这样做会爆空间，我们发现空间主要消耗在了内层节点上，一个可行的做法是只有当访问到内层的一个节点，才去建立它的左儿子和右儿子，这样就可以通过所有测试数据了。时间复杂度 $O(n+m\log^2 n)$。

3111：[Zjoi2013]蚂蚁寻路

容易观察出题目就是要选出一个图形，这个图形是凸凹交错的，第一处是凸的，之后的每处都和前面的相反，题目要求包含 k 个凹的，我们记 $p=2k+1$。我们考虑枚举这个图形最下面的一条边，记最下面是第 u 行，$dp[i][j][k]$ 表示第 k 个小矩形的右上角坐标为 (i,j) 蚂蚁能获得的最大值，不妨只考虑 k 为偶数的情况，k 为奇数时与偶数的情况是同理的，则有 $dp[i][j][k]=\max(dp[i'][j'][k-1]+Sum(i,j'+1,u,j))$，$i'<i$，$j'<j$。其中 Sum 表示一个子矩阵的和。直接做时间复杂度为 $O(n^3m^2p)$。在转移的时候维护一些前缀的最大值，即可做到每次 $O(1)$ 转移，时间复杂度变成 $O(n^2mp)$，可以通过所有测试数据。

3112：[Zjoi2013]防守战线

首先列出线性规划方程，记第 i 个点放 x_i 个塔，比如样例有：

$x_2 + x_3 \geq 1$
$x_1 + x_2 + x_3 + x_4 + x_5 \geq 4$
$x_3 + x_4 + x_5 \geq 2$

最小化目标函数 $X = x_1 + 5x_2 + 6x_3 + 3x_4 + 4x_5$。

考虑构造对偶方程：

$y_2 \leq 1$
$y_1 + y_2 \leq 5$
$y_1 + y_2 + y_3 \leq 6$
$y_2 + y_3 \leq 3$
$y_2 + y_3 \leq 4$

最大化目标函数 $Y = y_1 + 4y_2 + 2y_3$。

对方面的方程添加辅助变量使之成为等式，其中 $z_i \geq 0$：

$y_2 + z_1 = 1$
$y_1 + y_2 + z_2 = 5$
$y_1 + y_2 + y_3 + z_3 = 6$
$y_2 + y_3 + z_4 = 3$
$y_2 + y_3 + z_5 = 4$

然后在最后面添加一个辅助等式 $0 = 0$，并把上面得到的所有等式，用其中相邻的两项，拿下面的减去上面的得到新的 $n+1$ 个等式有：

$y_2 + z_1 = 1$
$y_1 + z_2 - z_1 = 4$
$y_3 + z_3 - z_2 = 1$
$-y_1 + z_4 - z_3 = -3$
$z_5 - z_4 = 1$
$-y_2 - y_3 - z_5 = -4$

我们发现每个 y_i 和 z_i 都在上面的等式中出现两次且一次为正一次为负，这就像一个网络中的流量平衡，因此可以用费用流求出最优方案。

建立附加源点 S，附加汇点 T。

对于第 i 个等式，记等号右侧常量为 c，若 c 为正，则有 S 向 i 连容量为 c 费用为 0 的边，若 c 为负，则由 i 向 T 连容量为 $-c$ 费用为 0 的边。

若 y_i 在第 i 个等式中出现为正，在第 j 个等式中出现为负，则由 i 向 j 连容量为正无穷费用为题目要求的安置个数 D_i 的边。

若 z_i 在第 i 个等式中出现为正，在第 j 个等式中出现为负，则由 i 向 j 连容量为正无穷费用为 0 的边。

求上面这个网络流图的最大费用最大流，由对偶原理知，最大费用值就是题目所要求的最小费用。

3114: Uva12546 Lcm Pair Sum

找规律做就可以了，答案就是 $\prod_{i=1}^{n}\left(\sum_{j=0}^{p_i} a_i^j + p_i \times a_i^{p_i}\right) + N$ 。

3122: [Sdoi2013]随机数生成器

对于 $A=0$ 进行特判输出，对于 $A=1$ 使用扩展欧几里得求出，对于 $x_1 = B = 0$ 进行特判输出，去掉这三种情况以后，下文的所有算式中分母都不会为 0，我们考虑更通用的做法。

根据 $x_i = Ax_{i-1} + B$，由待定系数法或者特征根解得通项公式为 $x_i = A^{i-1}\left(x_1 + \frac{B}{A-1}\right) - \frac{B}{A-1}$（显然当 $i=1$ 时此公式也成立）。记 P 为题目给的模数，对上面的分式我们通过逆元将其变为整式有 $C = \frac{B}{A-1} = B \times (A-1)^{P-2}$，则通项公式变成 $x_i = A^{i-1}(x_1 + C) - C$ 。我们所求为最小的 i 使得 $x_i \equiv T \pmod P$，即： $A^{i-1}(x_1 + C) - C \equiv T \pmod P$，通过移项整理得到： $A^{i-1} \equiv \frac{T+C}{x_1+C} \pmod P$，令 $S = \frac{T+C}{x_1+C} = (T+C) \times (x_1+C)^{P-2}$，则我们所求为最小的 i 使得 $A^{i-1} \equiv S \pmod P$（i 为正整数）。我们考虑求出最小的 x 使得 $A^x \equiv S \pmod P$，则 $x+1$ 就是答案。令 $x = i + Mj$，其中 $M = \lceil \sqrt{P} \rceil$，则 $A^{i+Mj} \equiv S \pmod P$，也就是说 $A^i \equiv \frac{S}{A^{Mj}} \equiv S \times \left(A^{Mj}\right)^{P-2} \pmod P$，我们考虑处理 $S \times \left(A^{Mj}\right)^{P-2}$ 对 P 取模的值，易知这样的值最多只有 $\lceil \sqrt{P} \rceil$ 个，然后把这些值存起来并排序，对于左面的 A^i，只需要进行二分查找即可得到答案。时间复杂度 $O\left(\sqrt{P} \log P\right)$。

3123: [Sdoi2013]森林

对于初始的每个连通分量，在每个连通分量内任意选择一个根节点，使无根树变为有根树。我们考虑维护每个连通分量内，每个点到这个连通分量根节点的每个数出现的次数，显然点 p 的信息与 p 的父亲的信息有大量重复，因此我们可以考虑对初始点权进行离散化以后使用函数式线段树维护。这样我们就得到了每个点到达所在连通分量根节点中每个数的出现次数。对于一个询问操作 p，q，k，我们预处理每个点的倍增父亲，这样就可以快速求出 p 和 q 的 LCA，然后在线段树内二分答案即可，计算方式是 p 到根节点的数量加上 q 到根节点的数量减去二倍 LCA 到根节点的数量加上 LCA 本身这个值（因为 LCA 被多减了一次）。

接下来我们考虑加边操作，对于一次加边 p，q，我们不妨设 p 所在的连通分量内的节点个数小于 q 所在的连通分量内的节点个数，这样我们就可以使用启发式合并的方式来合并 p 和 q 所在的两棵子树，合并的方式是将 p 设为 p 所在连通分量的根节点，然后把 p 这个点挂在 q 的下面，这样我们就把 p 所在的整棵子树都挂在了 q 的下面，接下来我们暴力重建 p 所在的子树每个节点的函数式线段树即可，然后再维护 p 所在子树每个节点的倍增父亲。注意回收一下 p 所在子树每个节点的函数式线段树节点，否则内存不够用。

由于使用启发式合并，因此加边复杂度为均摊单次 $O(\log^2 n)$，询问复杂度为单次 $O(\log n)$。

3124: [Sdoi2013]直径

我们考虑计算每条边有多少条直径经过它，记经过次数最大值为 R，则所有经过次数为 R 的边都必须在直径

上（因为任意两条直径不可能完全独立，重叠部分至少要是一个点，否则可以构造出更长的直径）。我们首先求出这棵树的直径记为 Q。对于每个点 i，我们求出 i 的子树中到 i 距离的最大值，次大值，最大值的方案数，次大值的方案数。然后求出不在 i 的子树中（就是 i 先走到 i 的父亲，然后在不回头的情况下随便走）的点到 i 的最大距离以及方案数。在 i 的子树中的情况可以通过一次自底向上的 TreeDP 求出，不在 i 的子树中的情况可以通过 i 的父亲转移而来，具体实现有很多细节，注意一下就好了。对于每条边(a,b)（其中 a 是 b 的父亲）有多少条直径经过它的计算方法是，如果这条边的长度加上 a 往上走的最大距离加上 b 往下走的最大距离不等于 Q 则经过次数为 0，否则根据乘法原理，经过次数就是往上走的方案数乘往下走的方案数。时间复杂度 $O(n)$。

3129：[Sdoi2013]方程

我们发现限制一共有上界和下界两种，我们首先不考虑任何一种限制，设所有未知数的和为 m，共有 n 个未知数，由隔板法易知方程组的解个数为 $C(m-1,n-1)$。接下来我们考虑所有下界，我们发现对于一个下界 $A_i>=x$，实际上我们可以把 m 减掉 $x-1$，这样下界就变成了 $A_i>=1$，而 $A_i>=1$ 大于等于 1 就相当于没有下界，因此我们把 m 减掉所有下界减一的和。由于上界的限制数量很少，我们可以暴力用容斥原理解决。所以时间复杂度为 $O(2^{n1}\times$组合数复杂度$)$。由于取模的数不是质数，我们可以考虑将其分解质因数以后用中国剩余定理合并答案，注意组合数计算的一些预处理过程，以免耽误时间。

3130：[Sdoi2013]费用流

根据贪心易知后手一定是将整个 P 的代价都放在流量最大的一条边上，接下来我们就是要让网络中流量最大的边最小化。考虑使用二分答案控制最大流量，然后看计算出的网络流等不等于最初的网络流即可。

3131：[Sdoi2013]淘金

记 $f(x)$表示 x 的各位数字之积，我们发现 $f(x)$不会含有除 2,3,5,7 以外的质因子，因此我们可以考虑预处理出 1 到 10^{12} 共有多少种不同的 $f(x)$，经过尝试发现只有不到 8,600 个。接下来我们考虑记录每个 $f(x)$在 1 到 n 中的出现次数，也就是统计 1 到 n 有多少个数的各位数字之积等于 $f(x)$。我们考虑预处理每个大区间，所谓大区间是指 [0,9]，[00,99]，[000,999]，[00..0(12 个 0),99..9(12 个 9)]，每个区间内每个 $f(x)$的出现次数。处理完这些以后 1 到 n 每个数的出现次数就可以很容易地用数位统计的方式求出。我们记第 i 个 $f(x)$的出现次数为 $t(i)$，题目所求为 $t(i)\times t(j)$的前 K 大值的和，这个问题很经典了，用堆就可以解决了。

3142：[Hnoi2013]数列

我们设原序列依次为 $a[1]$，$a[2]$，……，$a[k]$。定义差分序列 $b[i]=a[i+1]-a[i]$。其中 $1\le i<k$。我们发现对于一组 $b[i]$，我们需要满足 $a[1]+\sum b[i]\le n$。则 $1\le a[1]\le n-\sum b[i]$。由于 $m(k-1)<n$，因此 $\sum b[i]$ 的取值一定小于 n，也就是说对于任意一组 $\sum b[i]$ 都对应至少一组解，当 $\sum b[i]=x$ 时，对应解是数量为 $n-x$。因此我们只需要求出 $\sum\left(n-\sum b[i]\right)$。每个 $b[i]$的取值范围都是$[1,m]$，易知答案为 $m^{k-1}\left(n-\frac{(m+1)(k-1)}{2}\right)$。由于分母那个值并不一定是 2 的倍数，我们可以把取模的数先乘 2，然后再把答案除以 2 即可。

3143: [Hnoi2013]游走

我们记 $x[i]$表示点 i 的经过次数，$d[i]$点 i 的度，则可以列出方程 $x[i]=\sum\frac{x[j]}{d[j]}$，其中 i 和 j 之间有边且 j 不等于 n，特别的 $x[1]=\sum\frac{x[j]}{d[j]}+1$，其中 1 和 j 之间有边且 j 不等于 n。使用高斯消元解上面的方程组，接下来我们考虑计算一条边的期望经过次数。对于一条边 k 连接 i 和 j，则期望经过次数为 $E[k]=\frac{x[i]}{d[i]}+\frac{x[j]}{d[j]}$。然后我们把 $E[k]$按照从小到大的顺序排列，接下来只要贪心就好了，最大的 $E[k]$边权为 1，次大的 $E[k]$边权为 2，……，最小的 $E[k]$边权为 m。时间复杂度 $O(n^3+m\log m)$。