第九届北京师范大学程序设计竞赛解题报告

题号	标题	提交情况	用户情况
A	美女来找茬	45/132	45/46
B	XsugarX 的疯狂按键识别	41/95	40/41
C	删格游戏	25/62	25/32
D	商品装箱	5/33	5/11
E	侦察任务	0/33	0/4
F	A Simple But Difficult Problem	19/109	19/35
G	传送阵	33/151	33/46
H	左手定则	21/113	19/31
I	无爱编号	23/112	23/32
J	金坷垃之舞	8/62	6/11
K	Chromatron++	0/10	0/3

先总结一下这次校赛吧，这次题目对于 ACMer 来说可能偏于简单，从排名上也能看出来，有五只队伍现场都过了 9 题。但是这个题目难度我们认为作为北师大的校赛还是比较合适的，最后的排名也很漂亮。由于之前有说法，校内 AK 的冠军队伍，每人有 1000 元的奖金，然后这个奖金是出题组负担……然后由于对于校内大牛队实力的恐惧，所以，我们最后加入了 K 题……前面的题目，有不少题目（CDFG）被人水过去了，数据不够强……我们以后会加倍注意。

A 美女来找茬

简单签到题，只需要找出两矩阵中值相差大于 5 的点的坐标(x, y)中 x 和 y 分别的最大值和最小值即可。

B XsugarX 的疯狂按键识别

简单题，找位置最早出现的技能，因为不会有一个技能是另一个的前缀，所以用 strstr 即可。

C 删格游戏

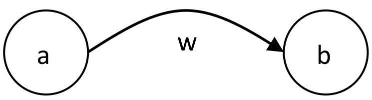
由于是非偏博弈，所以初始状态必定不是必胜态就是必败态。对于所有可能的初始局面（1*1 除外），假设先手必败，那么先手无论怎么拿，下一个都是必胜态。考虑如下情况，先手拿最右上角的方格，那么之后的状态就是个必胜态，但是，我们可以观察到，对于这个必胜态，后手随意怎么拿，先手都可以在上一轮取同样位置的方格达到同样的状态，也就是说，后手不能拿到一个必败态！所以假设不成立，对于所有初始局面（1*1 除外）先手必胜！

但是这题数据没有把 1000*1000 都出一遍，而是随机了不多的数据……所以让一些队伍胡猜过了，有些队伍猜了其他的必败情况，但是正好数据中没有……

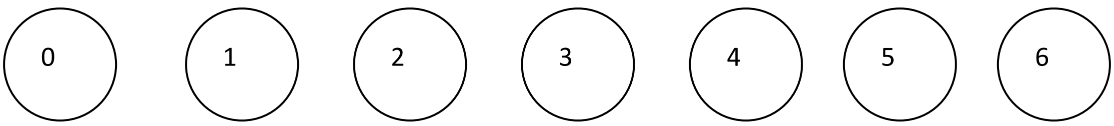
D 商品装箱

此题由于数据范围较小，所以有队伍用了 DP 方法把它水过……鸭梨很大……
无解的情况可分为两种：
(1)所有的数皆可以构造即有一个数是 1
(2)所有数的最大公约数>1(因为这样就只能构造出一种数)
而有解的情况
1、首先将 L1-Ln 排序（由小到大）
2、将所有的数，按除以 L1 的余数分类（余数从 R0-Rm），求出除以 L1 余数分别是 R0 到 Rm 的最小数字，这些最小数都能够有 L1-Ln 构成。这些最小数中的最大数字减去 L1 就是不能构成的最大数字。
如 L1=7 L2=9 L3=11
能够构成的数字除以 L1 余数为 0 的最小数字是 7；不可能有比它再小的数字了。
能够构成的数字除以 L1 余数为 1 的最小数字是 22；不可能有比它再小的数字了。
能够构成的数字除以 L1 余数为 2 的最小数字是 9；不可能有比它再小的数字了。
能够构成的数字除以 L1 余数为 3 的最小数字是 31；不可能有比它再小的数字了。
能够构成的数字除以 L1 余数为 4 的最小数字是 11；不可能有比它再小的数字了。
能够构成的数字除以 L1 余数为 5 的最小数字是 33；不可能有比它再小的数字了。
能够构成的数字除以 L1 余数为 6 的最小数字是 20；不可能有比它再小的数字了。
选出能够构成最小数字中的最大值 33，33 – L1 为 26 就是最大的不能构成的数字。

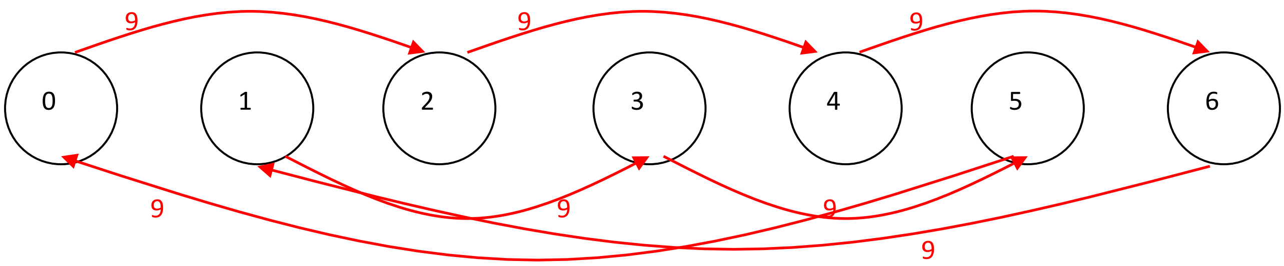
为了每个余数的最小值我们可以利用图论的计算最短路的方法。
图中的点表示每个余数 R0-Rm 即余数 0，1，2，3，4，5，6。
图中的边



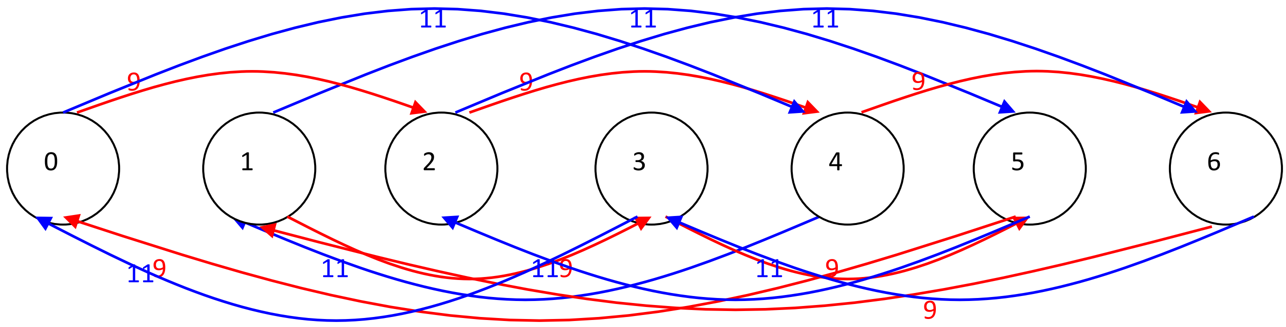
表示从余数是 a 的通过加 W 可以构成余数是 b 的。
初始状态



利用 L2 可以构成的图为



利用 L3 可以构成的为



E 侦察任务

通过给定的视野向量(vx,vy,vz)，四棱锥底面的 2 个边长以及与原点的距离，算出底面的四个顶点，就可以得到四棱锥五个面。轰炸机的飞行轨迹本质上是一条以初始位置为起点的射线，将它与四棱锥五个面判交，如果存在交点(px,py,pz)，计算(sx,sy,sz)+t*(rx,ry,rz) = (px,py,pz)中的 t 值，其中(sx,sy,sz)是轰炸机初始位置，(rx,ry,rz)是速度向量。如果跟 5 个多边形都没有交点，那么轰炸机不会落入视野范围，如果曾落入视野范围，则计算 t_{max}-t_{min}，如果得数严格大于警觉延迟时间，那么侦察兵发现敌机，否则会把它当作一只鹰。

此题数据十分强，手工构造了很多数据，最后没有队伍通过，虽然在我们意料之外，但是也算情理之中。最后校外提交的程序也就差了那么几组手工构造的样例。

手工数据：

发出警报：

// 从底面一点出发,沿矩形对角线往对点飞出

0.00 0.00 1.00 1.00 1.00 5.00 1.99

0.50 0.50 5.00 -0.50 -0.50 0.00

// 从底面外一点出发，沿视野方向飞入底面,从观察原点飞出

1.0 1.0 1.0 2.0 2.0 5.0 1.0

5.0 5.0 5.0 -1.0 -1.0 -1.0

// 从底面外一点出发，沿视野方向飞入底面,从观察原点飞出

1.0 1.0 1.0 2.0 2.0 5.0 2.00

5.0 5.0 5.0 -1.0 -1.0 -1.0

// // 从底面外一点出发，沿视野方向飞入底面,从观察原点飞出

1.0 1.0 1.0 2.0 2.0 5.0 2.88

3.6 3.6 3.6 -1.0 -1.0 -1.0

进入视野未警觉：

// 从底面一点出发,沿矩形对角线往对点飞出

0.00 0.00 1.00 1.00 1.00 5.00 2.01

0.50 0.50 5.00 -0.50 -0.50 0.00

// 从底面一点出发,沿矩形对角线往对点飞出

0.00 0.00 1.00 1.00 1.00 5.00 2.00

0.50 0.50 5.00 -0.50 -0.50 0.00
// 从底面外一点出发，沿视野方向飞入底面,从观察原点飞出
1.0 1.0 1.0 2.0 2.0 5.0 4.99
5.0 5.0 5.0 -1.0 -1.0 -1.0
// 从底面外一点出发，沿视野方向飞入底面,从观察原点飞出
1.0 1.0 1.0 2.0 2.0 5.0 4.99
3.6 3.6 3.6 -1.0 -1.0 -1.0
// 从观察原点正上方位置出发,沿视野方向飞行,与底面正上方的棱擦边飞过
1.00 0.00 0.00 2.00 2.00 5.00 0.01
0.00 1.00 0.00 1.00 0.00 0.00
// 从观察原点正上方位置出发,沿视野方向飞行,与底面正上方的棱擦边飞过
1.00 0.00 0.00 2.00 2.00 5.00 0.01
0.00 1.00 0.00 5.00 0.00 1.00

视野范围外：

// 沿着与底面平行的路线在四棱锥外飞过
0.00 0.00 1.00 1.00 1.00 4.99 1.00
0.50 0.50 5.00 -0.50 -0.50 0.00
// 从观察原点正上方位置出发,从四棱锥底面矩形正上方飞过
1.00 0.00 0.00 2.00 2.00 5.00 0.01
0.00 1.00 0.00 1.00 0.01 0.00
// 从观察原点正上方位置出发,从四棱锥底面矩形正上方飞过
1.00 0.00 0.00 2.00 2.00 5.00 0.01
0.00 1.00 0.00 5.00 0.00 1.01

F A Simple But Difficult Problem

由于这题的m不够大，所以出现了和我们预想的解法难度差很多的做法……这种做法如下：
由于

$$\mathrm{x}^k\%m = (a\%m)^k\%m$$

因此有

$$\begin{aligned}
&(1^k + 2^k + \cdots + n^k)\%m \\
&= \Big((1^k + 2^k + \cdots + m^k) + ((m+1)^k + (m+2)^k + \cdots + (m \times 2)^k) + \cdots \\
&+ ((t \times m + 1)^k + t * (m+2)^k + \cdots + n^k)\Big)\%m \\
&= \Big((1^k + 2^k + \cdots + m^k) \times (t-1) + (1^k + 2^k + \cdots + (n\%m)^k)\Big)\%m \\
&= \left((1^k + 2^k + \cdots + m^k) \times \frac{n}{m} + (1^k + 2^k + \cdots + (n\%m)^k)\right)\%m
\end{aligned}$$

标程解法：

对n^k做一个变形

$$n^k = A_n^k + \delta_k(n)$$

其中

$$A_n^k = n(n-1)\cdots(n-k+1), \qquad \delta_k(n) = n^k - A_n^k$$

变化一下A_n^k

$$A_n^k = \frac{1}{k+1}[(n+1)-(n-k)]n(n-1)\cdots(n-k+1)$$

变成

$$A_n^k = \frac{1}{k+1}[(n+1)n(n-1)\cdots(n-k+1) - n(n-1)\cdots(n-k+1)(n-k)]$$

也就是

$$A_n^k == \frac{A_{n+1}^{k+1} - A_n^{k+1}}{k+1}$$

所以

$$\sum_{i=1}^{n} n^k = \sum_{i=1}^{n} A_n^k + \sum_{i=1}^{n} \delta_k(n) = \frac{A_{n+1}^{k+1}}{k+1} + \sum_{i=1}^{n} \delta_k(n)$$

而

$$\delta_k(n) = n^k - A_n^k = n^k - \sum_{j=0}^{k} \alpha_k^{(j)} n^j$$

显然有$\alpha_k^{(k)} = 1$，$\delta_k(n)$是$k-1$次的多项式

$$\delta_k(n) = n^k - A_n^k = -\sum_{j=0}^{k-1} \alpha_k^{(j)} n^j$$

对其求和得到

$$\sum_{i=1}^{n} \delta_k(n) = -\sum_{i=1}^{n}\sum_{j=0}^{k-1} \alpha_k^{(j)} n^j = -\sum_{j=0}^{k-1}\sum_{i=1}^{n} \alpha_k^{(j)} n^j = -\sum_{j=0}^{k-1} \alpha_k^{(j)} \sum_{i=1}^{n} n^j$$

其中$\alpha_k^{(j)}$可以展开A_n^k为多项式来得到，

$$A_n^{k-1} = \sum_{j=0}^{k-1} \alpha_{k-1}^{(j)} n^j, \qquad A_n^k = \sum_{j=0}^{k} \alpha_k^{(j)} n^j$$

$$A_n^k = A_n^{k-1}(n-k) = (n-k)\sum_{j=0}^{k-1} \alpha_{k-1}^{(j)} n^j$$

$$A_n^k = \sum_{j=0}^{k-1} \alpha_{k-1}^{(j)} n^{j+1} - \sum_{j=0}^{k-1} k\alpha_{k-1}^{(j)} n^j = \sum_{j=1}^{k} \alpha_{k-1}^{(j-1)} n^j - \sum_{j=0}^{k-1} k\alpha_{k-1}^{(j)} n^j$$

于是

$$\mathrm{A_n^k} = -k\alpha_{k-1}^{(0)} + \sum_{j=1}^{k-1}\left[\alpha_{k-1}^{(j-1)} - k\alpha_{k-1}^{(j)}\right] n^j + \alpha_{k-1}^{(k-1)} n^k$$

比较系数得到如下的递推式

$$\alpha_k^{(j)} = \begin{cases} -k\alpha_{k-1}^{(0)} & j = 0 \\ \alpha_{k-1}^{(k-1)} & j = k \\ \alpha_{k-1}^{(j-1)} - k\alpha_{k-1}^{(j)} & 0 < j < k \end{cases}$$

初始条件为

$$\alpha_0^{(0)} = 0$$

$\sum_{i=1}^{n} n^j$可以用$n = 0,1,\dots,k-1$时候的结果计算得到。

整个算法的复杂度为$O(k^2)$。

G 传送阵

仔细读题后其实问题就是找到三个数让其和能够被另一个数 m 整除。
最暴力的办法是直接三层循环找到满足条件编号最小的三个数。（标准解法需要降维，在比赛过程中，数据考虑不周，导致暴力能够通过，表示抱歉）
标准解法：
a1,a2,a3 三个数的和能够被 m 整除 <=> a1%m, a2%m, a3%m 三个数的和能被 m 整除
以上两个关系可以互推，所以实际上所有的数可以预先模去 m，而因为 m 仅仅为 10000，所以无论 dp，还是其他解法都可以解决问题。
本人写的代码是把所有模去后的值做一个映射。枚举两层，可以求得满足条件的第三个数的大小，这样再查看是否可以映射到某个位置。时间复杂度为 n^2
因为要求输出编号最小的三个数，所以多记录一些数据和进行一些判断操作。具体判断请见标程。

H 左手定则

简单题，由于题意描述有一点不清楚，所以导致有队伍 WA 了好几次，对此表示抱歉，空地的情况标程的做法是先往左下右上的方向走四个格，而有队伍是上左下右，所以导致错误。做法很简单，就是按左前右后的顺序判断，模拟即可，需要记录一个 100*100*4 的矩阵，防止出现死循环。

I 无爱编号

简单 DP。
DP[i][0]表示前 i 位均不包含 4 和 13 且最后一位不是 1 的编号数量。
DP[i][1]表示前 i 位均不包含 4 和 13 且最后一位是 1 的编号数量。
显然 DP[0][0]=1,DP[0][1]=0。
又有 DP[i+1][0] = DP[i][0]*8 + DP[i][1]*7
DP[i+1][1]=DP[i][0]+DP[i][1]
O(N)计算一下即可。

J 金坷垃之舞

一开始我们找到了 O(N^2)的算法，并将这道题定位为难题。但在验证数据的过程中被出题组一神牛以 O(N^3)枚举+剪枝的算法秒掉…剪枝的方法比较直观且效果很好(运行时间是原先标程的一半)，而我们又没能找出可以成功卡掉这种算法的数据，所以将这道题重新定位为中等偏下题目。

两种算法的预处理是相同的：

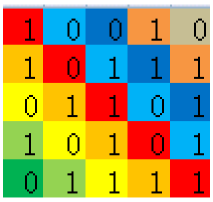
先以 O(N^2)的方法算出这个格子本身及往上下左右有多少个连续的 1，依次记作 U[i][j],D[i][j],L[i][j],R[i][j]。记 A[i][j]=min(D[i][j],R[i][j])，B[i][j]=min(U[i][j],L[i][j])。显然如果以(i,j)点为左上角存在一个边长为 k 的正方形的话必有 A[i][j],B[i+k][j+k]>=k。

首先介绍剪枝的算法：

记当前最优值为 best，然后枚举一个点为左上角(记枚举到的点为(i,j))，之后枚举边长，枚举区间为 best+1 到 A[i][j]，验证是否存在这样的矩形，如存在则更新 best。best 为最终结果。

O(N^2)的方法：

首先注意到验证正方形存在性时查看的是 A[i][j]和 B[i+k][j+k]，即是说需要查看的点都是在同一条斜线上，因此将全部的点按所在斜线分成 2N-1 个集合，每个集合单独处理。



如图中每种颜色为一个集合。在处理一个集合时先将格子重新编号（编号为 0,1,2,…,m-1），数组 a 和 b 对应每个数在原先位置时 A、B 的值。则重新编号后 i,j 两个点能构成正方形的条件为 a[i],b[j]>=j-i+1（j>=i）。此不等式等价于 i >= j-b[j]+1 (式 1)且 j<=i+a[i]-1(式 2)。

首先枚举 i，对于每一个 i 得到数对(i+a[i]-1, i)，数对(x,y)表示当 j<=x 时，j 与 i=y 满足式 2。

然后枚举 j，为了得到最大边长，当 j 固定时我们希望 i 尽可能小，同时 i 和 j 需要满足式 1 与式 2。

从小到大枚举 j，当 j=0 时，查看之前得到的全部数对(x,y)，如果 x>=0 则 j=0 与 i=y 满足式 2，于是我们得到一堆对于 j=0 满足式 2 的可行的 i。在满足式 2 的同时我们需要满足式 1，因此我们只需快速找到不小于 j-b[j]+1 的最小的可行的 i 即可。当枚举 j 到 x+1 时，如发现存在一数对(x,y)，则应将 y 从 i 的可行值中删除。于是问题转化为快速查找全部可行值中不小于某个值的最小的数和快速删除可行值的问题。

在这里我用并查集模拟链表。记并查集数组为 u，起始 u[i]=i+1,u[m]=m。在一开始统计数对时若发现 k 为可行值则令 u[k]=k。查找不小于 x 的最小的可行值的结果为 find(x)。删除 x 直接另 u[x]=x+1。于是查找、删除都在 O(1)的时间内解决。全程复杂度为 O(N^2)

相比剪枝的方法，并查集的方法代码长、较难理解、执行效果也不好，我猜测现场过掉本题的队伍应该都是用剪枝的方法过掉…原本也不太想写并查集的方法了，奈何负责人强烈要求只得啰嗦这么一堆…如果找到了能够卡掉枚举+剪枝的方法（其改进版为用随机枚举左上角代替 for 枚举左上角）的数据欢迎交流……

K Chromatron++

此题是防 AK 题目，只需要按照题目给定的规则，把所有的道具模拟一遍即可，注意要把合成光拆开处理比较合适，另外注意可能出现无限回转的路线，所以也要记录状态，或者直接限制 BFS 的节点数量。

题意描述比较难理解……所以我把游戏直接给出了下载链接，大家肯定玩的比较开心吧……