POJ3070【基础】

题目大意：

已知斐波那契数列可以通过矩阵来进行计算：

[F(n+1) Fn   ]   [1 1]^n

[Fn   F(n-1)]   =   [1 0]

，其中 Fn 表示斐波那契数列的第 n 项，F0=0。给定一个正整数 n，求出 Fn 的后四位数字。

输入：

有若干组测试数据，每一组测试数据有一个整数 n（1<=n<=10^9），当 n=-1 时测试数据结束。

输出：

每一组测试数据输出一行，包含一个整数，即 Fn 的后四位数字。

题解：

这是一题裸的矩阵快速幂。显然，当 n 足够大时，我们不能傻乎乎地做 n 次的矩阵乘法，所以需要用到快速幂：设我们要求的是 A^x，则将 x 化为 2 进制，然后依次做 A，A^2，(A^2)^2，……对 bin(x)中为 1 的位乘上对应次数的矩阵即可。举个例子：求 A^13，则只需要求 A^8×A^4×A^1 即可，而计算则只需要一路将 A^i 平方即可。

另外有一点需要注意的是，一般做快速幂的时候都会超过系统数据类型的范围，因此我们一般取其模 M 的结果。这个 mod M 的操作在每一步中都要进行，以免范围溢出。每一步中进行的正确行易证：(ak+b)(ck+d) mod m=(b+d) mod m。

这题里我还写了一些矩阵的其他操作，权作是一个完整的矩阵快速幂及相关操作的模板。

POJ3318【基础】

题目大意：

给出三个 n*n（1<=n<=500）的矩阵 A、B 和 C，求 A*B 是否等于 C。

输入：

第一行有一个整数 n，接下来有三个 n*n 的矩阵。

输出：
如果 A*B=C，则输出一行“YES”，否则输出一行“NO”。

题解：
虽然题目页面那里提示说 O（n^3）的算法不能通过，但是我自己写了一个朴素的直接相乘判断的 1800+ms 压着过了。另外不知道为什么我试着用整数读入外挂但是 WA 了。
网上有一种做法，即加一个行向量 R，则若 A*B=C 则 R*A*B=R*C。R 随机生成一个即可。那么复杂度将下降到 O（n^2）。不过实际测试了一下，这种方法也是只能跑到 1000+ms。不知道那些 79ms 的代码是怎么跑出来的……

POJ1830【基础】

题目大意、输入输出见原题（原题是中文描述的）。

题解：
对于每个灯有动它和不动它 2 种状态，用 1,0 表示。
构造列向量 X=[x1,x2....xn],Xi=0 或者 1。系数矩阵 B[b1,b2...bn]，变换矩阵 A[n][n]; 假设 A*X=B，那么 bi 等于 A 的第 i 行元素和 X 对应相乘，那么我们可以这样来设置 A[i][j]: 如果第 j 个灯操作会影响 i,那么 A[i][j]=1。
如果 i 的初始化状态和目标状态不同，那么 b[i]=1。因为 b[i]等于 A 的第 i 行元素和 X 对应相乘。那么也就是说第 i 个灯最后的状态是不是会变是由 X 决定的，因为 b[i]=sigma(a[i][j]*X[j])，这里的求和视为模 2 运算也就是异或,所以第 i 个灯最后的状态和初始比是不是变了是由每一个和 i 有关的灯决定的，这也就是为什么 j 会影响 i 设 A[i][j]=1 的原因。也就是为什么要设 b[i]是初始状态和最终状态的异或。

POJ1222【基础】

题目大意：
给出一个 5 行 6 列的灯泡矩阵，这些灯泡一开始有些是亮的有些没亮，如果人工操作改变一个灯泡的状态则会连带改变其上下左右（如果有的话）的灯泡的状态。求需要改变哪些灯泡的状态，另最后所有的灯泡都不亮。

输入：
第一行有一个整数 T 表示有多少组测试数据。接下来有 T 个 5 行 6 列的矩阵，1 表示这个位置的灯泡亮了，0 表示这个位置的灯泡没亮。

输出：
每一组测试数据先输出一行“PUZZLE #测试数据编号”，然后输出一个 5 行 6 列的矩阵，表示这个位置的灯泡被操作了，0 表示这个位置的灯泡没被操作。

题解：
POJ 1830 的变种，只是把灯泡固定到了 30 个而已，构造方程组的方式没有变。有一题几乎是完全一样的，POJ 1861，这里就不另写题解了。

POJ3233【基础】

题目大意：
给出一个 n*n 的矩阵 A（$1<=n<=30$），以及幂次 k（$1<=k<=10^9$），$S=A+A^2+\cdots\cdots+A^k$，求 S 的各个元素模 m（$1<=m<=10000$）后的余。

输入：
第一行有三个整数 n、k、m，接下来有一个 n*n 的矩阵，每一行的两个整数之间用空格分隔。

输出：
一个 n*n 的矩阵，格式同输入时的格式。

题解：
首先 A^x 可以用矩阵快速幂求出来（具体可见 poj 3070）。其次可以对 k 进行二分，每次将规模减半，分 k 为奇偶两种情况，如当 k=6 和 k=7 时有：

k=6 有 $S(6)=(1+A^3)*(A+A^2+A^3)=(1+A^3)*S(3)$

k=7 有 $S(7)=A+(A+A^4)*(A+A^2+A^3)=A+(A+A^4)*S(3)$

这是朴素的做法。由于递归调用耗时较多，所以跑的时间各有不同，我跑出来的是 140+ms。我看 Status 里有 0ms 过的，查了一下，惊奇地发现了这么一个东西：
我们看这个公式：（I 表示一个与 A 大小相等的单位阵）

$$设\ B=\begin{matrix} A & 0 \\ I & I \end{matrix}$$

$$\begin{matrix} A^2 & 0 \end{matrix}$$

$B^2=$

$A+I \qquad I$

多写几个，归纳法得到：

$$B^{(k+1)}=\begin{matrix} A^{(k+1)} & 0 \\ I+A+A^2+\cdots\cdots+A^k & I \end{matrix}$$

剩下的不用我多说什么了吧。用这种写法我跑出来的是 63ms，估计 0ms 是直接写数组和用一些如 memcpy 之类的函数来加快速度的。

POJ2947【中等】

题目大意：

有 n（1<=n<=300）种装饰物，m（1<=m<=300）个已知条件，每个已知条件的描述如下：

p start end

a1,a2......ap (1<=ai<=n)

第一行表示从星期 start 到星期 end 一共生产了 p 件装饰物(工作的天数为 end-start+1+7*x,加 7*x 是因为生产可能跨了很多周)，第二行表示这 p 件装饰物的种类(可能出现相同的种类，即 ai=aj)。规定每件装饰物至少生产 3 天，最多生产 9 天。问每种装饰物需要生产的天数。

输入：

有若干组测试数据。每一组测试数据第一行有两个整数 n 和 m，接下来有 m 个如上述描述的已知条件。当 n=m=0 时输入结束。

输出：

如果没有解，则输出“Inconsistent data.”，如果有多解，则输出“Multiple solutions.”，如果只有唯一解，则输出每种装饰物需要生产的天数。

题解：

设每一种产品需要的时间为 Xi，那么对于每一个条件，我们很容易建立一个同余方程，最后我们得到一个同余方程组：

a11*x1+a12*x2+……+a1n*xn 同余于 b1 (mod 7)

a21*x1+a22*x2+……+a2n*xn 同余于 b2 (mod 7)

……

am1*x1+am2*x2+……+amn*xn 同余于 bm (mod 7)
因此解的时候和一般的解方程有一些区别，那就是需要在消元操作的时候就要进行模余操作，因此我写了一个通分后相减的做法以方便模余操作。最后回代求解的时候，其实涉及到中国剩余定理（扩展欧几里德定理）的应用，但是由于 Xi 的范围已经被确定了，所以直接枚举即可。

POJ2065【中等】

题目大意：
给出一个素数 P(P<=30000)和一串长为 n 的字符串 str[]。字母'*'代表 0，字母 a-z 分别代表 1-26，这 n 个字符所代表的数字分别代表 f(1)、f(2)....f(n)。
定义: f (k) = sigma(0<=i<=n-1,ai*k^i (mod p)) (1<=k<=n,0<=ai<P)
求 a0、a1.....an-1。题目保证肯定有唯一解。

输入：
第一行有一个整数 T，表示有多少组测试数据。
接下来每一组测试数据有一行，每一行开头有一个整数 p，和一个长度不大于 70 的字符串 str。

输出：
每一组测试数据输出一行，包含 n 个整数，用空格分隔，即 a0 到 an-1 的值。

题解：
由题意可建立方程组
a0*1^0 + a1*1^1 + …… + an-1*1^(n-1) = f(1) (mod p)
a0*2^0 + a1*2^1 + …… + an-1*2^(n-1) = f(2) (mod p)
……
a0*n^0 + a1*n^1 + …… + an-1*n^(n-1) = f(n) (mod p)
注意最后求解的时候，因为 ai 已经确定了范围，而我们求解得到的是模 p 的值，所以除以系数的之前需要一直加 p（相当于同余方程组 ans[i] mod a[i][i]=0, ans[i] mod p=a[i][n]）直到为系数的整数倍为止。

POJ1166【中等】

题目大意：
规定一个钟有四种状态：0（时针指向 0 点）、1（时针指向 3 点）、2（时针指向 6 点）、3（时针指向 9 点）。现在有 9 个钟依次编号为 A～I，有 9 种操作 ABDE、ABC、BCEF、ADG、BDEFH、CFI、DEGH、GHI、EFHI 分别表示把对应编号

的钟的时针顺时针拨动 90 度。给出 9 个钟时针的初始状态，求最少要执行多少次何种操作，才能令所有的钟的时针指向 0。

输入：
有一个 3×3 的矩阵，依次表示第 A～I 个钟的初始状态。

输出：
按编号升序输出所有操作的编号，中间用一个空格分隔，比如 3 3 4 6 7 9。

题解：
如果我在考场上遇到这题，我第一反应肯定是暴力搜索，因为每一种操作只可能做 0～3 次，多了就没有意义了，因此总复杂度 4^9=262144 绝对可以在 1 秒内跑出结果。然而现在要用高斯消元来做。
由于钟转四次回到原位，所以还是一题同余方程。构造矩阵如下，第 i 行第 j 列是 1 表示第 i 个钟会被第 j 种操作改变状态：
1,1,0,1,0,0,0,0,0
1,1,1,0,1,0,0,0,0
0,1,1,0,0,1,0,0,0
1,0,0,1,1,0,1,0,0
1,0,1,0,1,0,1,0,1
0,0,1,0,1,1,0,0,1
0,0,0,1,0,0,1,1,0
0,0,0,0,1,0,1,1,1
0,0,0,0,0,1,0,1,1
然后根据读入，计算出每个钟需要被改变的状态数（即 4-当前状态值） mod 4 的结果，作为最后一列。
不知道为什么在解矩阵的时候不能选最大行和当前行交换，我在这里挂了很久，上网查了一下也没有一个明确的解释。解出来以后直接枚举每一种操作的次数（从 0 到 3）就可以得到结果了。

POJ3735【中等】

题目大意：
已知有 n 只喵星人，开始时每只喵星人有花生米 0 颗，先有一组操作：
由下面三个中的 k 个操作组成：
g i 　　给第 i 只喵星人一颗花生米
e i 　　让第 i 只喵星人吃掉它拥有的所有花生米
s i j 将喵星人 i 与猫咪 j 的拥有的花生米交换
将上述操作重复 m 次后，问每只猫咪有多少颗花生米？

输入：
有若干组测试数据。每一组测试数据第一行有三个整数 n，m，k（1<=n,k<=100，1<=m<=10^9）。接下来有 k 行每一行有一条操作，格式如上述。n=m=k=0 时测试数据结束。

输出：
每一组测试数据输出一行，包含 n 个空格分隔的整数，即第 1 到第 n 只喵星人最后拥有的花生米数量。

题解：
参见《Matrix67：十个利用矩阵乘法解决的经典问题》
http://blog.sina.com.cn/s/blog_680c5cd50100khvp.html
这题需要构造一个矩阵来解决这 m 次重复的操作，因为矩阵相乘可以用快速幂解决。下面以样例数据来进行说明。
一开始所有的喵星人都是没有花生米的，所以我们构造一个行向量 peanut=[0 0 0 1]，最后需要有多一个 1，原因后面会说。
然后我们对 k 次操作，构造一个矩阵 opt。并且规定 peanut*opt 得到的行向量为各喵星人的花生米数量。
首先，在没有任何操作的情况下，opt 显然是单位阵：
[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[0,0,0,1]
如果我们需要给某一只 x 喵星人一颗花生米，那么就在最后一行第 x 列的位置加 1，比如：
g 1：给 1 号 1 颗花生米：
[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[1,0,0,1]
中间还有：
g 2
[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[1,1,0,1]

g 2
[1,0,0,0]
[0,1,0,0]
[0,0,1,0]
[1,2,0,1]
这个时候 peanut*opt 由于 peanut 的最后一列是 1，所以就体现出了给第一只喵星人+1 的操作了。并且由于加花生的操作永远是在 opt 矩阵的最下面一行进行的，opt 矩阵的最右边一列永远不会被改变，所以 peanut 的最后一列永远是 1，永远可以体现加花生的操作。

交换操作，那么则交换 opt 矩阵的对应两列：
s 12：交换第 1,2 列：
[0,1,0,0]
[1,0,0,0]
[0,0,1,0]
[2,1,0,1]
因为一开始的时候，opt 的前 n 行 n 列是一个单位阵，而且不会被加花生操作改变数值，所以由矩阵的初等变换可知，交换以后实际上对应着交换了列变量两个位置的值，即两只猫咪的花生数被交换了。

清零操作，直接将对应列清零，这个也很好理解，就不解释了。

这一题没有要求模余，我只能直接粗暴地上了__int64。

POJ3613【中等】

题目大意：
给出一幅无向图和源点汇点，求出从源点到汇点经过 N 条边（可以重复走）的最短路的长度。

输入：
第一行有四个整数 N、T、S、E，T 表示有 T（1<=T<=100）条边，源点的编号是 S，汇点的编号是 E。接下来有 T 行，每一行有三个整数 w、u、v（1<=w,u,v<=1000），表示在点 u 和点 v 之间有一条边的长度为 w。

输出：
一行，一个整数，即所求的最短路长度。

题解：

这就是《Matrix67：十个利用矩阵乘法解决的经典问题》中的第八题。另外为了减低空间使用，我们可以先对出现的点进行重新编号（离散化），然后再来求解。

POJ3150【中等偏上】

题目大意：
有一个 n 个元素组成的环（1<=n<=500），指定一种 d-step 操作（1<=d<=n/2）：每一个元素的值变为所有与其距离小于等于 d（包括自己）的 2d+1 个元素的和再模 m（1<=m<=10^6）的余。给出环上各元素的初始值，求对这一系列的操作进行 k 次（1<=k<=10^7）d-step 操作后各元素的值。

输入：
第一行有四个整数 n、m、d、k。第二行有 n 个空格分隔的整数，表示环上第 1 到第 n 个元素的初始值。

输出：
一行，有 n 个空格分隔的整数，表示操作后环上第 1 到第 n 个元素的最终值。

题解：
我们规定 n 个元素组成一个行向量 num=[a1 a2 ... an]，因此每一次的 d-step 操作应该对应一个 n*n 的矩阵 opt，opt 第 i 行第 j 列对应的 1/0 表示第 j 个数要/不要乘上第 i 个数然后再模余，并且规定用 opt 右乘 num。以第一个测试数据为例，操作矩阵应该是：
[1,1,0,0,1]
[1,1,1,0,0]
[0,1,1,1,0]
[0,0,1,1,1]
[1,0,0,1,1]
很显然 opt 一开始的时候是对称的，理由很显然——ax 需要加上 ay，那么 ay 也要加上 ax。并且注意到，opt 矩阵的每一行/列都是上一行/列向右/下移动一位得到的(*)，这也很好理解。
因此我们要求的是 num*(opt^k)，所以我们先求出 opt^k。朴素的做法是直接用快速幂解决这个问题，并且老实说，如果我没有查题解，我也会直接这么写的。
然而牛逼的数学又来了。有大牛在做了若干次矩阵乘法后发现：
opt^2 =
[3,2,1,1,2]
[2,3,2,1,1]
[1,2,3,2,1]
[1,1,2,3,2]
[2,1,1,2,3]

opt^3 =
[7,6,4,4,6]
[6,7,6,4,4]
[4,6,7,6,4]
[4,4,6,7,6]
[6,4,4,6,7]
opt^4 =
[19,17,14,14,17]
[17,19,17,14,14]
[14,17,19,17,14]
[14,14,17,19,17]
[17,14,14,17,19]
无论 opt 的多少次方，得到的矩阵仍为对称阵，并且符合(*)的性质！大牛给出的解释是：
因为矩阵 A，B 具有 a[i][j]=A[i-1][j-1],B[i][j]=B[i-1][j-1]（i-1<0 则表示 i-1+n,j-1<0 则表示 j-1+n），
所以可以得出矩阵 C=a*b 也具有这个性质：
C[i][j]=sum(A[i][t]*B[t][j])=sum(A[i-1][t-1],B[t-1][j-1])=sum(A[i-1][t],B[t][j-1])=C[i-1][j-1] 。
因此我们只存储 opt 矩阵的第一行并且用这一行来计算就可以了！