# 2015年厦门大学ACM寒假集训六（0213）

我感觉这次还是挺难的，一共有6题（我找到之前一直要找的一道题了，就是第6题，这题很有启发意义，所以我放进来了）而且要过节了，给大家3天时间写吧。

这次要学的有：

1. 概率和数学期望：全概率公式、贝叶斯公式记一记就好了。加上做题的灵感。另外概率论这门课要好好学。
2. Pólya原理。建议直接记住burnside引理，这个引理是比Pólya原理更底层的证明。有了burnside引理，很容易得出Pólya的结论。

另外，这里还涉及置换的概念。

详细内容看：《Pólya原理及其应用》作者符文杰，先记住结论、做题吧，证明有时间再看。

1. 高斯消元法：《线性代数》这门课好好学就没问题了。

这里给出gauss-jordan 消元法的模板：

参数说明：对矩阵a进行消元，下标为a[1][1]..a[n][n+1],a[i][n+1]

const double eps=1e-8; //eps用来处理计算机精度有限的问题。

void gauss(double a[][maxn],int n){

int i,j,k,r;

for (i=1;i<=n;i++){

r=i;

for (j=i+1;j<=n;j++)

if (fabs(a[j][i])>fabs(a[r][i]))r=j;

if (fabs(a[r][i])<eps)continue;

if (r!=i)for (j=1;j<=n+1;j++)swap(a[i][j],a[r][j]);

for (k=1;k<=n;k++)if (k!=i)

for (j=n+1;j>=i;j--)

a[k][j]-=a[k][i]/a[i][i]\*a[i][j];

}

}

1. 这里我们来点高精度的吧，另外JAVA自带高精度 Bigint，用C的话，我给你们提供一份高精度模板吧。重载+-×/操作符使之可以按照正常数来做。

struct BigInt{

const static int mod = 10000;

const static int DLEN = 4; //压4位高精度

int a[600],len;

BigInt(){

memset(a,0,sizeof(a));

len = 1;

}

BigInt(long long v){

memset(a,0,sizeof(a));

len = 0;

do{

a[len++] = v%mod;

v /= mod;

}while(v);

}

BigInt operator +(const BigInt &b)const{

BigInt res;

res.len = max(len,b.len);

for(int i = 0;i < res.len;i++){

res.a[i] += ((i < len)?a[i]:0)+((i < b.len)?b.a[i]:0);

res.a[i+1] += res.a[i]/mod;

res.a[i] %= mod;

}

if(res.a[res.len] > 0)res.len++;

return res;

}

BigInt operator -(const BigInt &b)const{

BigInt res;

int x=0;

for (int i=0;i<len;i++){

res.a[i]=a[i]-x-b.a[i];

if (res.a[i]<0)res.a[i]+=mod,x=1;else x=0;

}

res.len=len;

while (res.a[res.len-1]==0 && res.len>1)res.len--;

return res;

}

BigInt operator \*(const BigInt &b)const{

BigInt res;

for(int i = 0; i < len;i++){

int up = 0;

for(int j = 0;j < b.len;j++){

int temp = a[i]\*b.a[j] + res.a[i+j] + up;

res.a[i+j] = temp%mod;

up = temp/mod;

}

if(up != 0)

res.a[i + b.len] = up;

}

res.len = len + b.len;

while(res.a[res.len - 1] == 0 &&res.len > 1)res.len--;

return res;

}

BigInt operator / (const int b){

BigInt res;

int x=0;

for (int i=len-1;i>=0;i--){

res.a[i]=(x\*mod+a[i])/b;

x=(x\*mod+a[i])%b;

}

res.len=len;

while (res.a[res.len-1]==0&&res.len>1)res.len--;

return res;

}

void output(){

printf("%d",a[len-1]);

for(int i = len-2;i >=0 ;i--)

printf("%04d",a[i]);

printf("\n");

}

};

给大家一个例子看怎么用吧：

int main(){

BigInt x;

x=123;

x=x+1;

x=x\*10;

x=x/5;

x.output();

return 0;

}

程序输出248，已经重载的操作符其用法可以当成普通的int来用。

当然你们可以实现自己的高精度，可以自己增加需要的功能，比如大小比较之类的。

最后：ACM求f(x)定积分时，经常用到自适应辛普森法（adaptied simpson rule）（除非你可以直接算出找到F(x)），大家了解一下，原理很简单。可自行搜索，核心代码就10行。

**Problem 1 uva 11021 Tribbles （概率计算）**

[**http://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&category=481&page=show\_problem&problem=1962**](http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=481&page=show_problem&problem=1962)

Description

You have a population of **k** Tribbles. Tribbles live for exactly one day and then die. before death, a single Tribble has the probability **Pi** of giving birth to **i** more Tribbles. What is the probability that after **m** generations, every Tribble will be dead?

**Input**  
The first line of input gives the number of cases, **N**. **N** test cases follow. Each one starts with a line containing **n**(1<=**n**<=1000), **k** (0<=**k**<=1000) and **m** (0<=**m**<=1000). The next **n** lines will give the probabilities **P0**, **P1**, ..., **Pn-1**.

**Output**  
For each test case, output one line containing "Case #**x**:" followed by the answer, correct up to an absolute or relative error of 10-6.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| **4**  **3 1 1**  **0.33**  **0.34**  **0.33**  **3 1 2**  **0.33**  **0.34**  **0.33**  **3 1 2**  **0.5**  **0.0**  **0.5**  **4 2 2**  **0.5**  **0.0**  **0.0**  **0.5** | **Case #1: 0.3300000**  **Case #2: 0.4781370**  **Case #3: 0.6250000**  **Case #4: 0.3164062** |

**Problem 2 Xmu 1300**

**Bt的BOSS**

**Description**

thinkpoet最近在玩一款游戏，经过若干小时的奋战，thinkpoet终于见到最终boss了，却发现最终boss太BT了。boss战的场地限制在一个矩形区域内，且该矩形被均匀划分成n\*m的小方格，这些小方格构成游戏中人物的最小移动单元。我们将第i行第j列的方格记为(i,j)，1<=i<=n, 1<=j<=m。thinkpoet发现每经过1个单位时间后boss就会释放出火焰或寒冰攻击。火焰攻击招式如下：  
1、随机选取一个方格(x1,y1)，在该方格放置火焰  
2、再随机选取一个方格(x2,y2)，同样在该方格放置火焰  
3、对所有min(x1,x2)<=x<=max(x1,x2),min(y1,y2)<=y<=max(y1,y2)的方格(x,y)，由于受到影响，这些格子上亦都会产生火焰  
寒冰攻击招式与火焰攻击招式类似，且每经过1个单位时间boss释放哪种攻击是随机的。注意，一个方格上出现火焰后是不会再被熄灭的，除非该方格又受到寒冰攻击的影响，此时该方格会恢复为正常的方格（也就是在寒冰招式攻击范围内，原来有火焰的方格受影响会恢复为正常方格）。反之，方格上出现寒冰的情况也是类似的。面对如此BT的boss，thinkpoet想知道经过t个单位时间后可走方格（即上面无火焰也无寒冰）的数目的期望值，以便设计出对抗boss的合理策略。为此，thinkpoet需要聪明的你来帮他这个忙。

**Input**

输入第一行是一个整数T（T<=10），表示测试数据的组数。  
每组输入分别是三个整数n,m和t，(1<=n,m<=100,1<=t<=10000)。

**Output**

输出可走方格的数目的期望值，请将结果保留到4位小数。

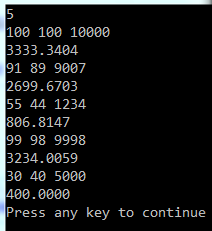
**Sample Input**

3  
1 1 1  
1 1 2  
2 2 10

**Sample Output**

0.0000  
0.5000  
1.3333

现在无法提交，所以我按照我的程序输入大点的数据、得出结果，大家可以参照一下：

**

*提示：这题对新手还是有点难度的，发挥你们的聪明才智吧。*

**Problem 3 uva 11542**

square

问题描述：

给出N个整数，从中选出1个或多个，使得选出的整数是完全平方数。一共有多少种选法？

比如{4,6,10,15}有3种选法{4},{6,10,15},{4,6,10,15}

输入格式：

第一行为T，表示测试数据组数。

每组数据包含两行：

第一行为n，第二行为n个数字a1..an, （0<ai<10^15），**不含>500的素因子**。

输出格式：

输出方案数，其大小保证不超过64位带符号数。

*提示：因数分解+高斯消元。*

**Problem 4 这里我们先考察下置换的概念。**

<http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=485&page=show_problem&problem=3255>

Leonardo的笔记本

问题描述：给出一个26个大写字母的置换B，求是否存在一个置换A，使得A^2=B

Sample Input

2

QWERTYUIOPASDFGHJKLZXCVBNM

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Sample Output

No

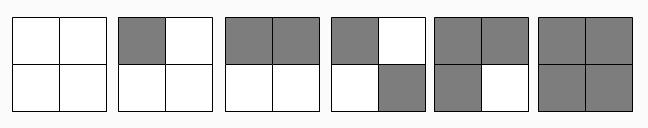
Yes

**Problem 5 Hdu 1812 polya原理。**

**Count the Tetris**

**Problem Description**

一个有N×N个格子的正方形棋盘，每个格子可以用C种不同颜色来染色，一共可以得到多少种不同的棋盘。如果一个棋盘，经过任意**旋转**，反射后变成另一个**棋盘**，这两个棋盘就是属于同一种棋盘。  
比如当N=C=2的时候，有下面六种不同的棋盘



现在告诉你N和C，计算有多少种不同的棋盘

**Input**

本题目包含多组测试，请处理到文件结束。  
每组测试数据包含两个正整数N和C(0<N,C,<31),分别表示棋盘的大小是N×N，用C种颜色来进行染色。

**Output**

对于每组测试，在一行里输出答案。

**Sample Input**

2 2

3 1

**Sample Output**

6

1

提示：1、这题需要用到高精度，建议先直接应用我上面给的高精度模板，我们重点学习理解polya原理。

2、如何确定有多少种置换？

题目允许旋转、反射两种操作，那么所有可以由这两种操作达到的状态，都算是一种置换。注意不能重复计算也不能遗漏，本身也算是一种。

反射：直白说就是左右翻转。

3、这题又有旋转又有反射。也就是对应8种置换、

（旋转0度、90度，180度,270度）×（不反射，反射）=4×2=8；

4、优化技巧：

1. 需要用到快速幂计算,a^b，否则超时。
2. 0<n,c<31，可以先把所有30×30中状态处理出来，然后在o(1)内解决询问。这个优化可以跑得更快。

另外，如果感觉有压力的，建议不用写高精度、不用快速幂优化，虽然AC不了，但是能理解Polya就达到目的了，除去高精度，代码也挺短的。

**Problem 6** [**http://uva.onlinejudge.org/index.php?option=com\_onlinejudge&Itemid=8&category=516&page=show\_problem&problem=4109**](http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=516&page=show_problem&problem=4109)

Joseph’s problem

问题描述：输入正整数n,k (n,k<=10^9),求 。

Sample Input

5 3

Sample Input

7

*提示：当一个一个算会超时的时候，我们就得想想有没有办法一类一类地算。*