涉及到组合数学的问题，首先是群的概念：

设G是一个集合，\*是G上的二元运算，如果(G,\*)满足下面的条件：

封闭性：对于任何a,b∈G,有a\*b∈G;

结合律：对任何a,b,c∈G有(a\*b)\*c=a\*(b\*c);

单位元：存在e∈G,使得对所有的a∈G,都有a\*e=e\*a=a;

逆元：对于每个元素a∈G,存在x∈G,使得a\*x=x\*a=e,这个时候记x为a-1，称为a的逆元，那么则称(G,\*)为一个群。

例：G={0,1,2,3,4....n-1}那么它在mod n加法下是一个群。

群元素的个数有限，称为有限群，且其中元素的个数称为阶，记为|G|,群元素的个数无限，称为无限群。

若对于群元素中的任意两个元素a,b都有ab=ba那么称G为交换群，简称Abel群。

=============================================================================================

置换：设X为一个有限集，π是X到X的一个--变换，那么称π是X上的一个置换。

例：设X={1,2,3,4....n},设π是X的一个变换，满足π：1->a1,2->a2,......n->an,其中a1,a2...an是X的一个排列，则称π是X上的一个置换。

可将π记为   1     2   ......   n

                  a1   a2   ......a n

同一置换用这样的表示法有n!种，但其对应的关系不变。

假设循环π只这样一个置换，满足π：a1->a2,a2->a3,.............ak->a1,但是对于其他元素保持不变，即：a->a,

可将π记为   a1     a2   ......   ak

                  a2   a3   ......  a1

称为k阶循环，K为循环长度。

每个置换都可以写成若干个互不相交的循环的乘积，且表示是唯一的.

如   1   2  3   4  5  6

       2   4   5  1  3  6    ，则可以表示为(124)(35)(6),置换的循环节数是上面的循环个数，上面的例题的循环节数为3.

=============================================================================================

定义：设G是有限集X上的置换群，点a,b∈X称为"等价"的，当且仅当，存在π∈G使得π(a)=b，记为a~b，这种等价条件下，X的元素形成的等价类称为G的轨道，它是集X的一个子集，G的任意两个不同的轨道之交是空集，所以置换群G的轨道全体是集合X的一个划分，构成若干个等价类，等价类的个数记为L。

**Zk (K不动置换类)：**设G是1…n的置换群。若K是1…n中某个元素，G中使K保持不变的置换的全体，记以Zk，叫做G中使K保持不动的置换类，简称K不动置换类。

**Ek(等价类)：**设G是1…n的置换群。若K是1…n中某个元素，K在G作用下的轨迹，记作Ek。即K在G的作用下所能变化成的所有元素的集合。.

这个时候有：|**Ek**|\*|**Zk**|=|G|成立(k=1,2,.....n)。

C(π)：对于一个置换π∈G,及a∈X，若π(a)=a，则称a为π的不动点。π的不动点的全体记为C(π)。例如π=(123)(3)(45)(6)(7)，X={1,2,3,4,5,6,7};那么C(π)={3,6,7}共3个元素。

Burnside引理：L=1/|G|\*(**Z1**+**Z2**+**Z3**+**Z4**+**......Zk**)=1/|G|\*(C(π1)+C(π2)+C(π3)+.....+C(πn))(其中k∈X,π∈G)。

Polya定理：设G={π1，π2，π3........πn}是X={a1，a2，a3.......an}上一个置换群，用m中颜色对X中的元素进行涂色，那么不同的涂色方案数为：1/|G|\*(mC(π1)+mC(π2)+mC(π3)+...+mC(πk)). 其中C(πk)为置换πk的循环节的个数。

polya定理求循环节个数代码模板：

const int MAX=1001;

#define CLR(arr,val) memset(arr,val,sizeof(arr))

int n,perm[MAX],visit[MAX];//sum求循环节个数,Perm用来存储置换,即一个排列

int gcd(int n,int m)

{ return m==0?n:gcd(m,n%m);

}

void Polya()

{ int pos,sum=0;

CLR(visit,0);

for(int i=0;i<n;i++)

if(!visit[i])

{ sum++;

pos=i;

for(int j=0;!visit[perm[pos]];j++)

{ pos=perm[pos];

visit[pos]=1;

}

}

return sum;

}

一般可以证明：当只有旋转的时候(顺时针或逆时针)，对于一个有n个字符的环，可顺时针或逆时针旋转几个位置，由于至少有n个置换，但是假设我顺时针旋转k个位置，他就等同于逆时针转动n-k个位置，假设一个置换为:G={π0，π1，π2，π3，π4，...，πn-1}，这个时候可以证明逆时针旋转k个位置时πk的循环节的个数为Gcd(n,k)，且每个循环的长度为L=n/gcd(n,i)。

例题1：[NYOJ 280(LK的项链)](http://acm.nyist.net/JudgeOnline/problem.php?pid=280)，LK的男朋友送给LK一盒有红、蓝、绿三种颜色的珠子，每种颜色珠子的个数都大于24，现在LK想用这一盒珠子穿出一条项链，项链上的珠子个数为n（0<=n<=24）,请你帮她计算一下一共可以用这一盒珠子可以穿出多少条不同的项链。通过旋转、翻转达到同一种状态的被认为是相同的项链。

涉及到旋转和翻转，上面已经说了旋转的情况，下面说下翻转的规律。

当n为奇数的时候，这个时候只有一种形式，假设经过某个顶点i与中心的连线为轴的翻转πi，共有n个，置换πi的形式如下，i保持不变：

πi：i->i，i+1->i-1，i+2->i-2，i+3->i-3.................i+n-1->(i-(n-1)+n)%n。

这个时候由对称性知，加上顶点i共有n个循环节数为(n+1)/2的循环群 。

当n为偶数时，有两种形式：

(1)、经过某个顶点与中心的连线为轴的翻转，有n/2个，这个时候和第一种为奇数的时候一样。

(2)、以顶点i和i+1的中点与中心的连线为轴翻转，共有n/2个：

πi:i->i+1，i-1->i+2，i-2->i+3，.................(i-j+n)%n->(i+j+1)%n。

这个时候共有n/2个循环节数(n+2)/2的循环群，和n/2个循环节数n/2的循环群。要特别注意0的情况，输出0即可。且由于对于输入不同的num均有2\*num中置换，所以结果应该是/(2\*num)。

#include<cstdio>

#include<cstring>

#include<cmath>

#include<algorithm>

using namespace std;

const int mod=9973;

const int maxn=100005;

#define LL int

struct mat

{

int mm;

int a[11][11];

}ma;

mat operator \*(const mat x,const mat y)

{

struct mat z;

int ii,jj,kk;

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

if(x.mm!=y.mm){z.mm=-1;return z;}

z.mm=x.mm;

for(ii=1;ii<=z.mm;ii++)

for(jj=1;jj<=z.mm;jj++)

{

for(kk=1;kk<=z.mm;kk++)

z.a[ii][jj]=(z.a[ii][jj]+x.a[ii][kk]\*y.a[kk][jj]);

z.a[ii][jj]=z.a[ii][jj]%mod;

}

return z;

}

mat operator ^(const mat x,const int yy)

{

int ii;

int y=yy;

struct mat p;

p.mm=x.mm;

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

for(ii=1;ii<=p.mm;ii++)p.a[ii][ii]=1;

struct mat t=x;

while(y)

{

if(y&1)p=p\*t;

t=t\*t;

y=y>>1;

}

return p;

}

int \_,\_\_;

int i,j,k,l,m,n;

LL gcd(LL a,LL b)

{

return b==0?a:gcd(b,a%b);

}

void tgcd(LL a,LL b,LL& d,LL& x,LL& y)

{

if(!b){d=a;x=1;y=0;}

else{tgcd(b,a%b,d,y,x);y-=x\*(a/b);}

}

LL pow\_mod(LL a,LL p,LL n)

{

if(p==0)return 1;

LL ans=pow\_mod(a,p/2,n);

ans=ans\*ans;

if(p%2==1)ans=ans\*a;

return ans%n;

}

int euler\_phi(int n)

{

int m=(int)sqrt(n+0.5);

int ans=n;

for(int i=2;i<=m;i++)

if(n%i==0)

{

ans=ans/i\*(i-1);

while(n%i==0)n=n/i;

}

if(n>1)ans=ans/n\*(n-1);

return ans%mod;

}

LL inv(LL a,LL n)

{

LL d,x,y;

tgcd(a,n,d,x,y);

return d==1?(x+n)%n:-1;

}

LL inv1(LL a,LL n)

{

return pow\_mod(a,n-2,mod);

}

int cou(int t,struct mat u)

{

struct mat tot=u^t;

int ii,ans;

ans=0;

for(ii=1;ii<=u.mm;ii++)ans+=tot.a[ii][ii];

return ans%mod;

}

void solve(struct mat q,int n)

{

long long int ans=0;

int ii;

for(ii=1;ii\*ii<=n;ii++)

{

if(n%ii!=0)continue;

ans=(ans+euler\_phi(ii)\*cou(n/ii,q))%mod;

if(n/ii!=ii)

ans=(ans+euler\_phi(n/ii)\*cou(ii,q))%mod;

}

ans=(ans\*inv(n,mod))%mod;

printf("%lld\n",ans);

}

int main()

{

scanf("%d",&\_\_);

for(\_=1;\_<=\_\_;\_++)

{

scanf("%d %d %d",&n,&m,&k);

ma.mm=m;

for(i=1;i<=m;i++)

for(j=1;j<=m;j++)ma.a[i][j]=1;

for(i=1;i<=k;i++)

{

scanf("%d %d",&j,&l);

ma.a[j][l]=0;

ma.a[l][j]=0;

}

solve(ma,n);

}

return 0;

}