Polya定理，Burnside引理 - CSDN博客

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。

Z_k (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Z_k，叫做G中使K保持不动的置换类，简称K不动置换类。

E_k(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E_k。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：|**E_k**|*|**Z_k**|=|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：$L=1/|G|*(Z_1+Z_2+Z_3+Z_4+......Z_k)=1/|G|*(C(\pi_1)+C(\pi_2)+C(\pi_3)+.....+C(\pi_n))$(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：$1/|G|*(m^{C(\pi_1)}+m^{C(\pi_2)}+m^{C(\pi_3)}+...+m^{C(\pi_k)})$. 其中$C(\pi_k)$为置换$\pi_k$的循环节的个数。

polya定理求循环节个数代码模板：

[cpp]view plaincopy

```cpp
constint 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
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={π_0，π_1，π_2，π_3，π_4，…，π_{n-1}}，这个时候可以证明逆时针旋转k个位置时π_k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)，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)。

[cpp]view plaincopy

```cpp
#include
#include
usingnamespace std;
#define __int64 long long
__int64 pow(int value,int num)
{   __int64 sum=1;
for(int i=1;i<=num;i++)
        sum*=value;
return sum;
}
int gcd(int n,int m)
{   return m==0?n:gcd(m,n%m);
}
__int64 Polya(int Color,int num)//长度为num具有Color中颜色的环形串的个数
{   __int64 sum=0;
for(int i=1;i<=num;i++)
        sum+=pow(Color,gcd(num,i));
if(num&1) sum+=num*pow(Color,(num+1)/2);
else sum+=(pow(Color,num/2+1)+pow(Color,num/2))*num/2;
return sum/2/num;
}
int main()
{   int num;
while(cin>>num,num!=-1)
    {   cout<<(num==0?0:Polya(3,num))<
    }
return 0;
}
```

当然如果你不知道翻转和旋转后的循环节为多少，我们可以自己构造置换，再利用上面求循环节的个数的模板求解即可，只是代码相对而言会比较的长。

[cpp]view plaincopy

```cpp
#include
#include
usingnamespace std;
constint MAX=50;
#define __int64 long long
#define CLR(arr,val) memset(arr,val,sizeof(arr))
int num,perm[MAX],visit[MAX];
```

```cpp
template<typename T>
__int64 Pow(T value,int num)
{    __int64 sum=1;
for(int i=1;i<=num;i++)
        sum*=value;
return sum;
}
__int64 Cycle()//total为循环节个数
{    __int64 pos,total=0;
    CLR(visit,0);
for(int i=0;i
if(!visit[i])
        {    total++;
            visit[pos=i]=1;
for(int j=0;!visit[perm[pos]];j++)//j记录当前循环节的长度
            {    pos=perm[pos];
                visit[pos]=1;
            }
        }
return total;
}
__int64 Polya()
{    __int64 sum=0;
for(int i=0;i
    {    for(int j=0;j//构造逆时针旋转i个位置形成的置换
            perm[j]=(i+j)%num;
        sum+=Pow(3,Cycle());
    }
if(num&1)
    {    for(int i=0;i
        {    for(int j=0;j//构造经过某点i与中心的连线为轴的翻转后形成的置换
                perm[(i+j)%num]=(i-j+num)%num;
            sum+=Pow(3,Cycle());
        }
    }
else
    {    for(int i=0;i
        {    for(int j=0;j//构造经过某点i与中心的连线为轴的翻转后形成的置换
                perm[(i+j)%num]=(i-j+num)%num;
            sum+=Pow(3,Cycle());
        }
for(int i=0;i
        {    for(int j=0;j//构造经过某点i与i+1的中点和中心的连线为轴的翻转后形成的置换
                perm[(i-j+num)%num]=(i+j+1)%num;
            sum+=Pow(3,Cycle());
        }
    }
return sum/2/num;
}
int main()
{    while(cin>>num,num!=-1)
        cout<<(num==0?0:Polya())<
return 0;
}
```

例题2：HDU 1257(最少拦截系统)，这个题目其实不属于Ploya定理，只是这个题目我也不知道放在哪里，但是我是根据求Ploya的循环节长度做的，所以顺便就贴在这里了。

[cpp]view plaincopy

```cpp
#include
#include
usingnamespace std;
constint MAX=100010;
#define CLR(arr,val) memset(arr,val,sizeof(arr))
int n,Arr[MAX],temp,visit[MAX];
```

```
int main()
{   while(cin>>n)
    {   int sum=0;
        CLR(visit,0);
for(int i=0;i
            cin>>Arr[i];
for(int i=0;i
if(!visit[i])
            {  sum++;
               visit[i]=1;
               temp=Arr[i];
for(int j=i+1;j
if(!visit[j]&&temp>Arr[j])
                   {   temp=Arr[j];
                       visit[j]=1;
                   }
            }
        cout<<
    }
return 0;
}
```

Polya定理题目总结：HDOJ 1812，2084，2647， POJ 2154，2409，2888等。