# 乘法逆元详解【费马小定理+扩展欧几里得算法】

# **乘法逆元**

## **何为乘法逆元？**

对于两个数a,p若gcd(a,p)=1则一定存在另一个数b，使得ab≡1(mod p)，并称此时的b为a关于1模p的乘法逆元。我们记此时的b为inv(a)或a−1。

举个例子：5×3≡1(mod14)，我们称此时的3为5关于1模14的乘法逆元。

## **如何求乘法逆元？**

### **方法一：费马小定理**

费马小定理：当有两数a,p满足gcd(a,p)=1时，则有ap≡a(mod p)。

变一下形：a\*ap−2≡1(modp)。是不是和上面的乘法逆元的定义是相似的？

所以，我们可以使用快速幂求出ap−2，即求出a的逆元。

### **方法二：扩展欧几里得算法**

由定义可知：ab≡1(mod p)，这个式子等价于已知a,p求一个二元一次不定方程ab=kp+1，移一下项得：ab−kp=1。这东西不是扩展欧几里得算法？

### **方法三：递推计算阶乘的逆元**

当我们要计算[一大](https://www.baidu.com/s?wd=%E4%B8%80%E5%A4%A7&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/qq_37656398/article/details/_blank)串连续的阶乘的逆元时，采用费马小定理或扩展欧几里得算法就有可能超时，所以我们必须采用一个更快的算法。

令fi=i!，则可得：

inv(fi+1)≡inv(fi\*(i+1))(mod p)

我们将(i+1)乘过去，则有：

inv(fi)≡inv(fi+1)\*(i+1)(mod p)

自然我们就得出递推式。

## **乘法逆元的作用？**

我们由费马小定理可得：a\*ap−2≡1(mod p)。

所以：

ap−2≡a−1≡1/a (mod p)

我们又知道模运算的乘法结合律：

b/a≡b\*a−1≡b\*ap−2(mod p)

所以我们可以知道：a除以一个数模p，等于a乘这个数的乘法逆元模p。

## **编程实现**

### **费马小定理：**

long long PowMod(long long a,int b) {

long long ret=1;

while(b) {

if(b&1)ret=ret\*a%Mod;

a=a\*a%Mod;

b>>=1;

}

return ret;

}

### **扩展欧几里得算法：**

long long extend\_gcd(long long a,long long b,long long &x,long long &y) {

if(a==0&&b==0)

return -1ll;

if(b==0)

{

x=1ll;

y=0ll;

return a;

}

long long d=extend\_gcd(b,a%b,y,x);

y-=a/b\*x;

return d;

}long long mod\_reverse(long long a,long long n) {

long long x,y,d=extend\_gcd(a,n,x,y);

if(d==1) {

if(x%n<=0)return x%n+n;

else return x%n;

} else return -1ll;

}

### **递推计算阶乘的逆元**

f[0]=1;for(int i=1;i<=N;i++)

f[i]=f[i-1]\*i%Mod;

inv[0]=1;

inv[N]=PowMod(f[N],Mod-2);

for(int i=N-1;i>0;i--)

inv[i]=inv[i+1]\*(i+1)%Mod;