乘法逆元

对于缩系中的元素，每个数a均有唯一的与之对应的乘法逆元x，使得ax≡1(mod n)
一个数有逆元的充分必要条件是gcd(a,n)=1，此时逆元唯一存在
逆元的含义：模n意义下，1个数a如果有逆元x，那么除以a相当于乘以x。

下面给出求逆元的几种方法：

1.扩展欧几里得

给定模数m，求a的逆相当于求解ax=1(mod m)
这个方程可以转化为ax-my=1
然后套用求二元一次方程的方法，用扩展欧几里得算法求得一组x0,y0和gcd
检查gcd是否为1
gcd不为1则说明逆元不存在
若为1，则调整x0到0~m-1的范围中即可

PS：这种算法效率较高，常数较小，时间复杂度为O(ln n)

[cpp] view plain copy

```cpp
typedef  long long ll;
void extgcd(ll a,ll b,ll& d,ll& x,ll& y){
    if(!b){ d=a; x=1; y=0;}
    else{ extgcd(b,a%b,d,y,x); y-=x*(a/b); }
}
ll inverse(ll a,ll n){
    ll d,x,y;
    extgcd(a,n,d,x,y);
    return d==1?(x+n)%n:-1;
}
```

2.费马小定理

在模为素数p的情况下，有费马小定理
a^(p-1)=1（mod p）
那么a^(p-2)=a^-1(mod p)
也就是说a的逆元为a^(p-2)

而在模不为素数p的情况下，有欧拉定理
a^phi(m)=1（mod m） (a⊥m)
同理a^-1=a^(phi(m)-1)

因此逆元x便可以套用快速幂求得了x=a^(phi(m)-1)

但是似乎还有个问题？如何判断a是否有逆元呢?

检验逆元的性质，看求出的幂值x与a相乘是否为1即可

PS:这种算法复杂度为O（log_2N）在几次测试中，常数似乎较上种方法大

当p比较大的时候需要用快速幂求解

[cpp] view plain copy

```cpp
typedef  long long ll;
ll pow_mod(ll x, ll n, ll mod){
    ll res=1;
    while(n>0){
        if(n&1)res=res*x%mod;
        x=x*x%mod;
        n>>=1;
    }
    return res;
}
```

当模p不是素数的时候需要用到欧拉定理

a^phi(p)≡**1**   **(mod p)**
a*a^(phi(p)-1)≡1   (mod p)
a^(-1)≡a^(phi(p)-1) (mod p)
所以
时间复杂度即求出单个欧拉函数的值
(当p为素数的时候phi(p)=p-1,则phi(p)-1=p-2可以看出欧拉定理是费马小定理的推广)
PS：这里就贴出欧拉定理的板子，很少会用欧拉定理求逆元

3.特殊情况

一:

当N是质数，
这点也很好理解。当N是质数，0 < a < N时，，则a肯定存在逆元。
而解出的就满足，故它是a的逆元。

在CF 696C,

求解就灰常方便了…
二:
求逆元一般公式(条件b|a)

ans=a/bmodm=amod(mb)/b

公式证明:

$$\frac{a}{b} \bmod k = d$$
$$\frac{a}{b} = kx + d$$
$$a = kbx + bd$$
$$a = (kb)x + bd$$
$$a \bmod kb = bd$$
$$\frac{a \bmod kb}{b} = d$$

PS:实际上a mod (bm)/b这种的对于所有的都适用，不区分互不互素,而费马小定理和扩展欧几里得算法求逆元是有局限性的，它们都会要求 a 与 m 互素，如果a与m不互素，那就没有逆元，这个时候需要a mod (bm)/b来搞（此时就不是逆元的概念了）。但是当a与m互素的时候，bm可能会很大，不适合套这个一般公式，所以大部分时候还是用逆元来搞

4.逆元打表

有时会遇到这样一种问题，在模质数p下，求1~n逆元 n< p（这里 p 为奇质数）。可以O(n)求出所有逆元，有一个递推式如下

$$inv[i] = (M - M/i) * inv[M\%i]\%M$$

它的推导过程如下，设 $t = M/i$ $k = M\%i$，那么

$$
\Rightarrow t * i + k \equiv 0(modM)
$$
$$
\Rightarrow -t * i \equiv k(modM)
$$

对上式两边同时除 $i * k$，进一步得到

$$
-t * inv[k] \equiv inv[i](modM)
$$

再把 t 和 k 替换掉，最终得到

$$
inv[i] = (M - M/i) * inv[M\%i]\%M
$$

初始化 $inv[1] = 1$，这样就可以通过递推法求出1->n模奇素数 p 的所有逆元了。

另外有个结论 $1 \rightarrow p$ 模 p 的所有逆元值对应 $1 \rightarrow p$ 中所有的数，比如 $p = 7$，那么 $1 \rightarrow 6$ 对应的逆元是 $1\ \ 4\ \ 5\ \ 2\ \ 3\ \ 6$。

[cpp] view plain copy

```cpp
typedef  long long ll;
const int N = 1e5 + 5;
int inv[N];

void inverse(int n, int p) {
    inv[1] = 1;
    for (int i=2; i<=n; ++i) {
        inv[i] = (ll) (p - p / i) * inv[p%i] % p;
    }
}
```

例题可以参考http://blog.csdn.net/acdreamers/article/details/8220787

来源：https://blog.csdn.net/guhaiteng/article/details/52123385