乘法逆元

原题地址
参 考

题目背景

展开

这是一道模板题

题目描述

给定 n, p 求 $1 \sim n$ 中所有整数在模 p 意义下的乘法逆元。

输入格式

一行两个正整数 n, p。

输出格式

输出 n 行，第 i 行表示 i 在模 p 下的乘法逆元。

输入输出样例

输入 #1 复制

```
10 13
```

输出 #1 复制

```
1
7
9
10
8
11
2
5
3
4
```

说明/提示

$1 \leq n \leq 3 \times 10^6, n < p < 20000528$

输入保证 p 为质数。

拓展欧几里得(单个查找，p可以为合数)

```cpp
void Exgcd(ll a, ll b, ll &x, ll &y) {
    if (!b) x = 1, y = 0;
    else Exgcd(b, a % b, y, x), y -= a / b * x;
}
int main() {
    ll x, y;
    Exgcd (a, p, x, y);
    x = (x % p + p) % p;
    printf ("%d\n", x); //x是a在mod p下的逆元
}
```

快速幂(单个查找，p必须为质数)

```cpp
ll quick_pow(ll a,ll b,ll mod){//a^b%mod
    ll ans=1,base=a;
    while(b){
        if(b&1) ans*=base,ans%=mod;
        base*=base,base%=mod;
        b>>=1;
    }
    return ans;
}
int main() {
    ll x = quick_pow(a, p - 2, p); //x为a在mod p意义下的逆元
}
```

线性递推(连续查找，p必须为质数)

```cpp
inv[1] = 1;
for(int i = 1; i < p; ++ i)
    inv[i] = (p - p / i) * inv[p % i] % p;
```

`AC` 代码

```cpp
//#pragma GCC optimize(2)
//std::ios::sync_with_stdio(0)
//clock_t st=clock();
#include<bits/stdc++.h>
#define abss(x) ((x)>(0)?(x):(-1)*(x))
#define maxs(a,b) ((a)>(b)?(a):(b))
#define mins(a,b) ((a)<(b)?(a):(b))
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)
#define ROF(i,a,b) for(int i=(a);i>=(b);--i)
#define mem(a) memset(a,0,sizeof(a))
const int INF (1<<30);
const int inf (-1<<30);
```

```
using namespace std;

const int maxn=3e6+7;
int inv[maxn];

int main(){
    int n,p;
    cin>>n>>p;
    inv[1]=1;
    cout<<"1\n";
    FOR(i,2,n){
        inv[i]=(long long)(p-p/i)*inv[p%i]%p;
        printf("%d\n",inv[i]);
    }
}
```

矩阵快速幂

[P3390 【模板】矩阵快速幂]

```
#include<bits/stdc++.h>
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)
#define ll long long
#define mem(a) memset(a,0,sizeof(a))
#define scan(a) scanf("%lld",&(a))
#define print(a) printf("%lld",a)
using namespace std;

const int maxn=105;
const int mod=1e9+7;
ll n,k;

struct matx{
    ll a[maxn][maxn];
    matx(){
        mem(a);
    }
    void unit(){
        FOR(i,1,n)
            a[i][i]=1;
    }
    matx operator *(const matx &b){
        matx c;
        FOR(k,1,n)
            FOR(i,1,n)
                FOR(j,1,n)
                    c.a[i][j]=(c.a[i][j]+a[i][k]*b.a[k][j]%mod)%mod;
        return c;
```

```
    }
    matx operator *=(const matx &b){
        *this=(*this)*b;
        return *this;
    }
};

matx pow(matx Ma,ll k){
    matx Mans;
    Mans.unit();
    do{
        if(k&1) Mans*=Ma;
        Ma*=Ma;
        k>>=1;
    }while(k);
    return Mans;
}

int main(){
    matx Ma;
    cin>>n>>k;
    FOR(i,1,n)
        FOR(j,1,n)
            scan(Ma.a[i][j]);
    matx Mans=pow(Ma,k);
    FOR(i,1,n){
        FOR(j,1,n)
            print(Mans.a[i][j]),putchar(' ');
        putchar('\n');
    }
    return 0;
}
```

快速幂 排列组合

快速幂

不带模数

```
ll qpow(ll a,ll b){//a^b
    ll ans=1,base=a;
    while(b){
        if(b&1) ans*=base;
        base*=base;
        b>>=1;
    }
    return ans;
}
```

带模数，循环式

```
ll qpow(ll a,ll b,ll p){//a^b%p
    ll ans=1,base=a;
    while(b){
        if(b&1) ans*=base,ans%=p;
        base*=base,base%=p;
        b>>=1;
    }
    return ans;
}
```

带模数，递归式

```
ll qpow(ll a,ll b,ll p){
    if(b==1) return a;
    ll t=qpow(a,b/2,p);
    t=t*t%p;
    if(b&1) t=t*a%p;
    return t;
}
```

排列组合

需要用到费马小定理

```
ll C(ll n,ll m,ll p){
    if(n<m) return 0;
    if(m>n-m) m=n-m;
    ll a=1,b=1;
    FOR(i,0,m-1){
        a=(a*(n-i))%p;
        b=(b*(i+1))%p;
    }
    return a*qpow(b,p-2,p)%p;
}
```

卢卡斯定理

[P3807 【模板】卢卡斯定理/Lucas 定理]

$Lucas$ 定理：对于质数 p，有
$\binom{n}{m} \bmod p = \binom{\lfloor n/p \rfloor}{\lfloor m/p \rfloor} \cdot \binom{n mod p}{m mod p} \bmod p$
等价于
$C(n,m)\%p = C(n/p,m/p) * C(n\%p,m\%p)\%p$
边界条件：当 $m=0$ 的时候，返回 1 。

`C(n,m)%p = C(n/p,m/p) * C(n%p,m%p)%p`

Code :

```
#include<bits/stdc++.h>
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)
typedef long long ll;
using namespace std;

ll qpow(ll a,ll b,ll p){//快速幂
    ll ans=1,base=a;
    while(b){
        if(b&1) ans*=base,ans%=p;
        base*=base,base%=p;
        b>>=1;
    }
    return ans;
}

ll C(ll n,ll m,ll p){//组合数
    if(n<m) return 0;
    if(m>n-m) m=n-m;
    ll a=1,b=1;
    FOR(i,0,m-1){
        a=(a*(n-i))%p;
        b=(b*(i+1))%p;
    }
    return a*qpow(b,p-2,p)%p;//费马小定理
}

ll Lucas(ll n,ll m,ll p){//卢卡斯定理
    if(m==0) return 1;
    return Lucas(n/p,m/p,p)*C(n%p,m%p,p)%p;
}

int main(){
    int T;cin>>T;
    while(T--){
        ll n,m,p;
        cin>>n>>m>>p;
        cout<<Lucas(n+m,m,p)%p<<endl;
    }
    return 0;
}
```

裴蜀定理

[原题地址-Luogu]

题目描述

[ ] 展开

给定一个包含 n 个元素的**整数**序列 A，记作 $A_1, A_2, A_3, \ldots, A_n$。

求另一个包含 n 个元素的待定**整数**序列 X，记 $S = \sum_{i=1}^{n} A_i \times X_i$，使得 $S > 0$ 且 S 尽可能的小。

输入格式

第一行一个整数 n，表示序列元素个数。

第二行 n 个整数，表示序列 A。

输出格式

一行一个整数，表示 $S > 0$ 的前提下 S 的最小值。

输入输出样例

输入 #1 复制

```
2
4059 -1782
```

输出 #1 复制

```
99
```

说明/提示

对于 100% 的数据，$1 \le n \le 20$，$|A_i| \le 10^5$，且 A 序列不全为 0。

裴蜀定理

对于整数`a,b`和正整数`x,y`，`ax+by=c`成立的**充要**条件是`gcd(a,b)%c=0`.

推论：`a,b`互质的**充要**条件是存在整数`x,y`，使`ax+by=1`.

拓展：对于`n`个整数`a1,a2,......,an`，`a1*x1+a2*x2+......+an*xn=S`成立的**充要**条件是`gcd(a1,a2,......an)%S=0`.

`AC` 代码

```
//#pragma GCC optimize(2)
//std::ios::sync_with_stdio(0)
//clock_t st=clock();
#include<bits/stdc++.h>
#define abss(x) ((x)>(0)?(x):(-1)*(x))
#define maxs(a,b) ((a)>(b)?(a):(b))
#define mins(a,b) ((a)<(b)?(a):(b))
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)
```

```
#define ROF(i,a,b) for(int i=(a);i>=(b);--i)
#define mem(a) memset(a,0,sizeof(a))
const int INF (1<<30);
const int inf (-1<<30);
using namespace std;

int main(){
    int n;
    cin>>n;
    int ans=0,t;
    FOR(i,1,n){
        scanf("%d",&t);
        t=abss(t);
        ans=__gcd(ans,t);
    }
    cout<<ans;
}
```

拓展欧几里得(exgcd)

```
void exgcd(int &x,int &y,int a,int b){
    //使用时exgcd(a,b,a,b)即可，无需全局变量，运行后a、b为一组解
    if(!b){
        x=1,y=0;
        return;
    }
    exgcd(x,y,b,a%b);
    int t;
    t=x,x=y,y=t-a/b*y;
}
```

压行版本

```
void exgcd(int &x,int &y,int a,int b) {
    if(!b) x=1,y=0;
    else exgcd(y,x,b,a%b),y-=a/b*x;
}
```

线性筛素数

```
//#pragma GCC optimize(2)
//clock_t st=clock();
#include<bits/stdc++.h>
#define abss(x) ((x)>(0)?(x):(-1)*(x))
#define maxs(a,b) ((a)>(b)?(a):(b))
#define mins(a,b) ((a)<(b)?(a):(b))
```

```
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)
#define ROF(i,a,b) for(int i=(a);i>=(b);--i)
#define mem(a) memset(a,0,sizeof(a))
const int INF (1<<30);
const int inf (-1<<30);
using namespace std;

const int maxn=1e8+7,maxm=6e6;
bool isPrime[maxn];
int Prime[maxm],cnt=0;

void GetPrime(int n){//数据范围[1,n]
    memset(isPrime,1,sizeof(isPrime));
    isPrime[1]=0;

    FOR(i,2,n){
        if(isPrime[i])//没被筛掉
            Prime[++cnt]=i;//i成为下一个素数

        for(int j=1;j<=cnt and i*Prime[j]<=n;j++){
            isPrime[i*Prime[j]]=0;
            if(i%Prime[j]==0) break;
        }
    }
}//素数被标记为1，合数被标记为0

int main(){
    int n,q,k;
    cin>>n>>q;
    GetPrime(n);
    while(q--){
        scanf("%d",&k);
        printf("%d\n",Prime[k]);
    }
    return 0;
}
```

卡特兰数

卡特兰数

Catalan 数列

H_0	H_1	H_2	H_3	H_4	H_5	H_6	...
1	1	2	5	14	42	132	...

递推关系的解：

`H(n) = C(2n,n)/(n+1)`

$$H_n = \frac{\binom{2n}{n}}{n+1}(n \geq 2, n \in \mathbf{N}_+)$$

关于 Catalan 数的常见公式：

`H(0) = H(1) = 1`,`H(n) = SUM(i=1->n){H(n-1)*H(n-i)}`

$H_n = \{\sum_{i=1}^{n} H_{i-1}H_{n-i} \quad n \geq 2, n \in \mathbf{N}_+ \; 1 \quad n = 0, 1$

`H(n) = H(n-1)*(4n-2)/(n+1)`

$H_n = \frac{H_{n-1}(4n-2)}{n+1}$

`H(n) = C(2n,n)-C(2n,n-1)`

$H_n = \binom{2n}{n} - \binom{2n}{n-1}$

第二类斯特林数

第二类斯特林数

定义：`S(n,k)`，表示将`n`个两两不同的元素，划分为`k`个互不区分的非空子集的方案数。

递推式：`S(n,k) = S(n-1,k-1) + k*S(n-1,k)`

边界：`S(0,0)=0`,`S(1~n,0)=1`

递推式：$\begin{Bmatrix} n \\ k \end{Bmatrix} = \begin{Bmatrix} n-1 \\ k-1 \end{Bmatrix} + k \begin{Bmatrix} n-1 \\ k \end{Bmatrix}$

边界：$\begin{Bmatrix} n \\ 0 \end{Bmatrix} = [n = 0]$。

通项公式：`S(n,k) = SUM(i=0->k){[(-1)^(k-i) * i^n] / [i! * (k-i)!]}`

通项公式：$\begin{Bmatrix} n \\ k \end{Bmatrix} = \sum_{i=0}^{k} \frac{(-1)^{k-i} i^n}{i!(k-i)!}$

欧拉函数（复变函数） 五边形数 分割函数

欧拉函数（复变函数）

$$\phi(x) = \prod_{k=1}^{\infty}(1 - x^k)$$

将这个式子展开

$$\prod_{k=1}^{\infty}(1-x^k)=\sum_{k=-\infty}^{\infty}(-1)^k x^{\frac{3k^2-k}{2}}$$

其中 x 的次数，$\frac{3k^2-k}{2}$，为广义五边形数

将得到的所有项按升幂排列，得到：

$$\phi(x)=1-x-x^2+x^5+x^7-x^{12}-x^{15}+\dots$$

分割函数

定义：将正整数 n 拆分为若干个正整数的和（允许同一个数使用多次，这里的拆分是无序的，即 1+2 和 2+1 等价）的方案数。

欧拉函数的倒数是**划分数（分割函数）**的**生成函数**：

$$\frac{1}{\varphi(x)}=\sum_{i=0}^{\infty}p(i)x^i=\prod_{i=0}^{\infty}\sum_{j=0}^{\infty}x^{ij}=\prod_{i=0}^{\infty}\frac{1}{1-x^i}$$

其中 $p(i)$ 为 i 的分割函数

有

$$\varphi(x)\times\frac{1}{\varphi(x)}=1\Leftrightarrow\left(\sum_{k=0}^{\infty}(-1)^k x^{\frac{k(3k\pm1)}{2}}\right)\left(\sum_{i=0}^{\infty}p(i)x^i\right)=1$$

展开得到递推式

$$p(k)-p(k-1)-p(k-2)+p(k-5)+p(k-7)-\dots=0$$

分割函数的代码实现 $O(n\sqrt{n})$

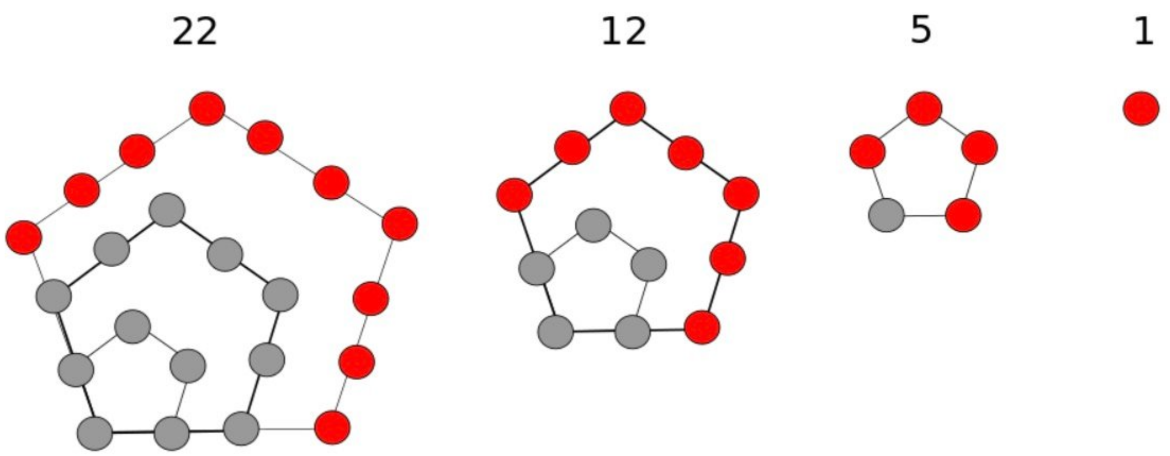
P6189 [NOI Online #1 入门组] 跑步

```
#include<bits/stdc++.h>
#define FOR(i,a,b) for(int i=(a);i<=(b);++i)
using namespace std;

const int maxn (1e5+7);
const int INF (1<<30);

long long f[maxn];

```

```
inline int a(int x){
    return (3*x*x-x)/2;
}

int main(){
    int n,p;
    cin>>n>>p;
    f[0]=1;
    FOR(i,1,n){
        FOR(j,1,INF){
            int x=a(j),y=a(-j);
            if(x<=i) f[i]=((f[i]+(j&1?1:-1)*f[i-x])%p+p)%p;
            //f(i)=f(i)+(-1)^(j+1)*f(i-x)
            if(y<=i) f[i]=((f[i]+(j&1?1:-1)*f[i-y])%p+p)%p;
            if(x>i or y>i) break;
        }
    }
    cout<<f[n];
    return 0;
}
```

五边形数

五边形数是能排成五边形的多边形数。



通项公式： $$p_n = \frac{3n^2 - n}{2}$$

广义五边形数的公式和五边形数相同，只是 n 可以为负数和零，**n 依序为0, 1, -1, 2, -2, 3, -3, 4...**，广义五边形数也可以用下式表示：

$$p_n = \frac{3n^2 \pm n}{2}$$

n 依序为0, 1, 2, 3, 4, 5, 6, 7, 8...

其产生的数列如下：

0, 1, 2, 5, 7, 12, 15, 22, 26...

欧拉函数（数论）

定义

欧拉函数，$\varphi(n)$，表示小于等于 n 且与 n 互质的数的个数。

欧拉函数的值

$$\varphi(n)=p_1^{k_1-1}p_2^{k_2-1}\cdots p_r^{k_r-1}(p_1-1)(p_2-1)\cdots(p_r-1),$$

等价形式

$$\varphi(n)=n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\cdots\left(1-\frac{1}{p_r}\right).$$

代码实现：

```
int euler_phi(int n) {
  int ans = n;
  for (int i = 2; i * i <= n; i++)
    if (n % i == 0) {
      ans = ans / i * (i - 1);
      while (n % i == 0) n /= i;
    }
  if (n > 1) ans = ans / n * (n - 1);
  return ans;
}
```

性质

1. 若 n 为质数，则 $\varphi(n)=n-1$。
2. 欧拉函数是积性函数。若 $\gcd(a,b)=1$，则 $\varphi(a\times b)=\varphi(a)\times\varphi(b)$。
3. $n=\sum_{d|n}\varphi(d)$。（$d|n$ 表示 n 能被 d 整除）
4. 若 $n=p^k$，p 为质数，则 $\varphi(n)=p^k-p^{k-1}$。
5. 欧拉定理：若 $\gcd(a,m)=1$，则 $a^{\varphi(m)}\equiv 1\pmod m$。
6. 拓展欧拉定理：

$$a^b\equiv\begin{cases}a^{b\bmod\varphi(p)}, & \gcd(a,\,p)=1\\ a^b, & \gcd(a,\,p)\neq 1,\,b<\varphi(p)\\ a^{b\bmod\varphi(p)+\varphi(p)}, & \gcd(a,\,p)\neq 1,\,b\geq\varphi(p)\end{cases}\pmod p$$

7. 费马小定理：

若 p 为素数，$\gcd(a,p)=1$，则 $a^{p-1}\equiv 1 \pmod{p}$。

另一个形式：对于任意整数 a，有 $a^p\equiv a \pmod{p}$。