

# 万能欧几里得算法

## 算法简介

一种  $O(\log n)$  计算形如  $f(n) = \sum_{i=1}^n \lfloor \frac{ai+b}{c} \rfloor$  的函数的算法。

## 算法原理

不妨考虑待计算函数就是  $f(n) = \sum_{i=1}^n \lfloor \frac{ai+b}{c} \rfloor$  同时假设  $b \perp c$  考虑按如下过程计算  $f(n)$

考虑在  $(0, n]$  范围内逐渐增加  $x$  当  $y = \frac{ax+b}{c}$  恰好等于某个整数时  $y \rightarrow y+1$  称为  $U$  操作。

当  $x$  恰好等于某个整数时有  $\text{ans} \rightarrow \text{ans} + y$  称为  $R$  操作。如果遇到整点，先进行  $U$  操作再进行  $R$  操作。

于是计算  $f(n)$  等价于处理由  $y = \frac{ax+b}{c} (x \in (0, n])$  产生的操作序列  $S$

先考虑如何得到这个操作序列，设  $g(a, b, c, n, U, R)$  表示由  $y = \frac{ax+b}{c} (x \in (0, n])$  产生的操作序列  $S$

定义字符串幂操作  $S^k = S^{k-1}S$

如果  $a \geq c$  显然  $R$  操作前面至少有  $\lfloor \frac{ac}{c} \rfloor$  个  $U$  操作。将这  $\lfloor \frac{ac}{c} \rfloor$  个  $U$  操作和一个  $R$  操作视为一个整体，于是有

$$g(a, b, c, n, U, R) = g(a \bmod c, b, c, n, U, U^{\lfloor \frac{ac}{c} \rfloor} R)$$

如果  $b \geq c$  显然有

$$g(a, b, c, n, U, R) = U^{\lfloor \frac{bc}{c} \rfloor} g(a, b \bmod c, c, n, U, R)$$

最后，如果  $c > \max(a, b)$  则考虑构造  $y = \frac{ax+b}{c}$  按  $y = x$  做对称变换得到的直线  $y = \frac{cx-b}{a}$

不难发现  $y = \frac{cx-b}{a}$  产生的序列和  $y = \frac{ax+b}{c}$  产生的序列大致是互补的，即  $U/R$  互补。

但要处理遇到整点的情况，此时两条直线都是先  $U$  再  $R$  不满足互补关系，考虑将  $y = \frac{cx-b}{a}$  整体向下偏移得到  $y = \frac{cx-b-1}{a}$

这样原先在  $y = \frac{cx-b}{a}$  先  $U$  再  $R$  的整点等价于  $y = \frac{cx-b-1}{a}$  先  $R$  在  $U$  于是整点也满足了互补性质。

但偏移也导致  $y = \frac{cx-b-1}{a}$  的边界发生了变化，考虑暴力处理边界，设  $m = \frac{an+b}{c}$  于是有

$$g(a, b, c, n, U, R) = R^{\lfloor \frac{c-b-1}{a} \rfloor} U g(c, (c-b-1) \bmod a, a, m-1, R, U) R^{\lfloor n - \lfloor \frac{c-b-1}{a} \rfloor \rfloor}$$

$\lfloor \frac{cm-b-1}{a} \rfloor$

边界条件为  $m=0$  此时说明只有  $R$  操作，于是  $g(a,b,c,n,U,R)=R^n$

假如不考虑字符串拼接等操作的复杂度，上述式子可以  $O(\log n)$  计算。

接下来考虑如何通过得到的字符串即操作序列来计算答案。事实上可以在拼接字符串时维护答案。

设  $h(S)$  表示字符串  $S$  对应的答案，两个串分别为

$S_1, S_2$   $dx = \text{cntR}(S_1), dy = \text{cntU}(S_1), n = \text{cntR}(S_2)$

$$h(S_1S_2) = h(S_1) + \sum_{i=1}^n (\lfloor \frac{ai+b}{c} \rfloor + dy) = h(S_1) + \sum_{i=1}^n (\lfloor \frac{ai+b}{c} \rfloor) + n \times dy = h(S_1) + h(S_2) + n \times dy$$

至于  $S^k$  可以利用快速幂计算。整体操作复杂度为  $T(a,c) = T(c \bmod a, a) + O(\log \frac{ca}{a})$

$T(a,c)$  产生的  $O(\log c) - O(\log a)$  可以和随后  $T(c \bmod a, a)$  产生的  $O(\log a) - O(\log (c \bmod a))$  部分抵消。

最终  $T(a,c) = O(\log c) - O(\log \gcd(a,c)) \sim O(\log c)$

## 算法例题

### 例题一

[洛谷p5170](#)

#### 题意

求

$$f(n) = \sum_{i=0}^n \lfloor \frac{ai+b}{c} \rfloor \quad g(n) = \sum_{i=0}^n (\lfloor \frac{ai+b}{c} \rfloor)^2 \quad h(n) = \sum_{i=0}^n i \lfloor \frac{ai+b}{c} \rfloor$$

#### 题解

设  $F, G, H$  分别表示序列  $S$  对应  $f, g, h$  的答案，于是有

$$\begin{aligned} F(S_1S_2) &= F(S_1) + \sum_{i=1}^n (\lfloor \frac{ai+b}{c} \rfloor + dy) = F(S_1) + F(S_2) + n \times dy \\ G(S_1S_2) &= G(S_1) + \sum_{i=1}^n (\lfloor \frac{ai+b}{c} \rfloor + dy)^2 = G(S_1) + \sum_{i=1}^n (\lfloor \frac{ai+b}{c} \rfloor)^2 + 2dy \sum_{i=1}^n \lfloor \frac{ai+b}{c} \rfloor + \sum_{i=1}^n (dy)^2 = G(S_1) + G(S_2) + 2dyF(S_2) + n \times (dy)^2 \\ H(S_1S_2) &= H(S_1) + \sum_{i=1}^n (i+dx) (\lfloor \frac{ai+b}{c} \rfloor + dy) = H(S_1) + \sum_{i=1}^n (i \lfloor \frac{ai+b}{c} \rfloor + i \times dy + dx (\lfloor \frac{ai+b}{c} \rfloor + dy)) \\ &= H(S_1) + H(S_2) + \frac{n(n+1)}{2} dy + dx F(S_2) + n \times dx dy \end{aligned}$$

最后这题的  $i$  是从  $0$  开始的，万能欧几里得算法的  $i$  是从  $1$  开始的，注意补上  $i=0$  的贡献。

```

const int mod=998244353;
struct Node{
    LL cntu,f,g,h;
    Node operator * (const Node &b)const{
        Node c;
        LL n=b.cntu,dx=cntu,dy=cntu;
        c.cntu=(cntu+b.cntu)%mod;
        c.cntu=(cntu+b.cntu)%mod;
        c.f=(f+b.f+dy*n)%mod;
        c.g=(g+b.g+dy*b.f*2+dy*dy%mod*n)%mod;
        c.h=(h+b.h+n*(n+1)/2%mod*dy+dx*b.f+dx*dy%mod*n)%mod;
        return c;
    }
};
Node quick_pow(Node n,int k){
    Node ans=Node{0,0,0,0,0};
    while(k){
        if(k&1)ans=ans*n;
        n=n*n;
        k>>=1;
    }
    return ans;
}
Node asgcd(int a,int b,int c,int n,Node su,Node sr){
    if(a>=c)
        return asgcd(a%c,b,c,n,su,quick_pow(su,a/c)*sr);
    int m=(1LL*a*n+b)/c;
    if(!m)
        return quick_pow(sr,n);
    else
        return quick_pow(sr,(c-b-1)/a)*su*asgcd(c,(c-
b-1)%a,a,m-1,sr,su)*quick_pow(sr,n-(1LL*c*m-b-1)/a);
}
Node cal(int a,int b,int c,int n){
    Node su=Node{0,1,0,0,0},sr=Node{1,0,0,0,0};
    return quick_pow(su,b/c)*asgcd(a,b%c,c,n,su,sr);
}
int main()
{
    int T=read_int();
    while(T--){
        int n=read_int(),a=read_int(),b=read_int(),c=read_int();
        Node ans=cal(a,b,c,n);
        ans.f=(ans.f+b/c)%mod;
        ans.g=(ans.g+1LL*(b/c)*(b/c))%mod;
        space(ans.f);space(ans.g);enter(ans.h);
    }
    return 0;
}

```

## 例题二

### LOJ#138

#### 题意

求

$$f(n) = \sum_{i=0}^n i^{\lfloor \frac{ai+b}{c} \rfloor}$$

#### 题解

设  $F(S, k_1, k_2)$  表示  $f(n) = \sum_{i=1}^n i^{\lfloor \frac{ai+b}{c} \rfloor}$  关于  $S$  操作序列的答案。

$$\begin{aligned} F(S_1 S_2, k_1, k_2) &= F(S_1, k_1, k_2) + \sum_{i=1}^n (i^{\lfloor \frac{ai+b}{c} \rfloor} + dx)^{\lfloor \frac{ai+b}{c} \rfloor + dy} \\ &= F(S_1, k_1, k_2) + \sum_{i=1}^n \sum_{t_1=0}^{\lfloor \frac{ai+b}{c} \rfloor} \sum_{t_2=0}^{\lfloor \frac{ai+b}{c} \rfloor + dy} \binom{\lfloor \frac{ai+b}{c} \rfloor}{t_1} \binom{\lfloor \frac{ai+b}{c} \rfloor + dy}{t_2} dx^{k_1 - t_1} dy^{k_2 - t_2} i^{t_1} \\ &= F(S_1, k_1, k_2) + \sum_{t_1=0}^{\lfloor \frac{ai+b}{c} \rfloor} \sum_{t_2=0}^{\lfloor \frac{ai+b}{c} \rfloor + dy} \binom{\lfloor \frac{ai+b}{c} \rfloor}{t_1} \binom{\lfloor \frac{ai+b}{c} \rfloor + dy}{t_2} dx^{k_1 - t_1} dy^{k_2 - t_2} F(S_2, t_1, t_2) \end{aligned}$$

对每个  $S$  维护矩阵  $F(S, i, j) (0 \leq i \leq k_1, 0 \leq j \leq k_2)$  于是可以实现  $O(k_1^2 k_2^2)$  信息合并。

边界  $F(L, i, j) = 0, F(R, i, j) = [j=0] (0 \leq i \leq k_1, 0 \leq j \leq k_2)$  另外注意对答案补上  $i=0$  的贡献。时间复杂度  $O(k_1^2 k_2^2 \log c)$

```
const int mod=1e9+7,MAXK=11;
int C[MAXK][MAXK],k1,k2;
struct Node{
    int cntr,cntu,f[MAXK][MAXK];
    Node(int cntr=0,int cntu=0){
        this->cntr=cntr;
        this->cntu=cntu;
        mem(f,0);
    }
    Node operator * (const Node &b)const{
        static int px[MAXK],py[MAXK];
        Node c;
        int dx=cntr,dy=cntu;
        px[0]=py[0]=1;
        _rep(i,1,k1)
            px[i]=1LL*px[i-1]*dx%mod;
        _rep(i,1,k2)
            py[i]=1LL*py[i-1]*dy%mod;
        c.cntr=(cntr+b.cntr)%mod;
        c.cntu=(cntu+b.cntu)%mod;
```

```

        _rep(i,0,k1)_rep(j,0,k2){
            c.f[i][j]=f[i][j];
            _rep(i2,0,i)_rep(j2,0,j)
c.f[i][j]=(c.f[i][j]+1LL*b.f[i2][j2]*C[i][i2]%mod*C[j][j2]%mod*px[i-
i2]%mod*py[j-j2])%mod;
        }
        return c;
    }
};
Node quick_pow(Node n,int k){
    Node ans=Node(0,0);
    while(k){
        if(k&1)ans=ans*n;
        n=n*n;
        k>>=1;
    }
    return ans;
}
Node asgcd(int a,int b,int c,int n,Node su,Node sr){
    if(a>=c)
        return asgcd(a%c,b,c,n,su,quick_pow(su,a/c)*sr);
    int m=(1LL*a*n+b)/c;
    if(!m)
        return quick_pow(sr,n);
    else
        return quick_pow(sr,(c-b-1)/a)*su*asgcd(c,(c-
b-1)%a,a,m-1,sr,su)*quick_pow(sr,n-(1LL*c*m-b-1)/a);
}
Node cal(int a,int b,int c,int n){
    Node su=Node(0,1),sr=Node(1,0);
    _rep(i,0,k1)
    sr.f[i][0]=1;
    return quick_pow(su,b/c)*asgcd(a,b%c,c,n,su,sr);
}
int main()
{
    C[0][0]=1;
    _for(i,1,MAXK){
        C[i][0]=1;
        _rep(j,1,i)
        C[i][j]=(C[i-1][j-1]+C[i-1][j])%mod;
    }
    int T=read_int();
    while(T--){
        int n=read_int(),a=read_int(),b=read_int(),c=read_int();
        k1=read_int(),k2=read_int();
        Node ans=cal(a,b,c,n);
        int base=1;
        _rep(i,0,k2){
            ans.f[0][i]=(ans.f[0][i]+base)%mod;
            base=1LL*base*(b/c)%mod;
        }
    }
}

```

```
    }  
    enter(ans.f[k1][k2]);  
}  
return 0;  
}
```

### 例题三

LOJ#6440

#### 题意

求

$$\sum_{i=1}^n A^i B^{\lfloor \frac{a_i+b}{c} \rfloor}$$

其中  $A, B$  为  $k \times k$  的矩阵。

#### 题解

$$\begin{aligned} F(S1S2) &= F(S1) + \sum_{i=1}^n A^{i+dx} B^{\lfloor \frac{a_i+b}{c} \rfloor + dy} \\ &= F(S1) + A^{dx} \left( \sum_{i=1}^n B^{\lfloor \frac{a_i+b}{c} \rfloor} \right) B^{dy} \\ &= F(S1) + A^{dx} F(S2) B^{dy} \end{aligned}$$

于是可以令  $F(S)$  维护  $(A^{cntR}, B^{cntU}, \text{ans})$  设  $E$  表示单位矩阵  $Z$  表示全  $0$  矩阵。

边界条件  $F(U) = (E, B, Z), F(R) = (A, E, A), F() = (E, E, Z)$  时间复杂度  $O(k^3 \log c)$

```
const int mod=998244353,MAXN=21;  
const __int128 One=1;  
int sz;  
struct Matrix{  
    int a[MAXN][MAXN];  
    Matrix(int type=0){  
        mem(a,0);  
        if(type){  
            _for(i,0,MAXN)  
                a[i][i]=1;  
        }  
    }  
    Matrix operator + (const Matrix &b) const{  
        Matrix c;  
        _for(i,0,sz)_for(j,0,sz)  
            c.a[i][j]=(a[i][j]+b.a[i][j])%mod;  
        return c;  
    }  
    Matrix operator * (const Matrix &b) const{
```

```

    Matrix c;
    _for(i,0,sz)_for(j,0,sz)_for(k,0,sz)
    c.a[i][j]=(c.a[i][j]+1LL*a[i][k]*b.a[k][j])%mod;
    return c;
}
};
struct Node{
    Matrix A,B,f;
    Node operator * (const Node &b)const{
        Node c;
        c.A=A*b.A;
        c.B=B*b.B;
        c.f=f+A*b.f*B;
        return c;
    }
};
Node quick_pow(Node n,LL k){
    Node ans;
    ans.A=Matrix(1);
    ans.B=Matrix(1);
    while(k){
        if(k&1)ans=ans*n;
        n=n*n;
        k>>=1;
    }
    return ans;
}
Node asgcd(LL a,LL b,LL c,LL n,Node su,Node sr){
    if(a>=c)
        return asgcd(a%c,b,c,n,su,quick_pow(su,a/c)*sr);
    LL m=(One*a*n+b)/c;
    if(!m)
        return quick_pow(sr,n);
    else
        return quick_pow(sr,(c-b-1)/a)*su*asgcd(c,(c-
b-1)%a,a,m-1,sr,su)*quick_pow(sr,n-(One*c*m-b-1)/a);
}
Matrix A,B;
Node cal(LL a,LL b,LL c,LL n){
    Node su,sr;
    su.A=Matrix(1);
    su.B=B;
    sr.A=A;
    sr.B=Matrix(1);
    sr.f=A;
    return quick_pow(su,b/c)*asgcd(a,b%c,c,n,su,sr);
}
int main()
{
    LL a=read_LL(),c=read_LL(),b=read_LL(),n=read_LL();
    sz=read_int();

```

```
_for(i,0,sz)_for(j,0,sz)
A.a[i][j]=read_int();
_for(i,0,sz)_for(j,0,sz)
B.a[i][j]=read_int();
Node ans=cal(a,b,c,n);
_for(i,0,sz){
    _for(j,0,sz)
    space(ans.f.a[i][j]);
    putchar('\n');
}
return 0;
}
```

## 参考资料

这两篇还有配图，方便理解。

[C3H5ClO的博客](#)

[ix35\\_的博客](#)

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:jxm2001:%E4%B8%87%E8%83%BD%E6%AC%A7%E5%87%A0%E9%87%8C%E5%BE%97%E7%AE%97%E6%B3%95&rev=1629457161](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:%E4%B8%87%E8%83%BD%E6%AC%A7%E5%87%A0%E9%87%8C%E5%BE%97%E7%AE%97%E6%B3%95&rev=1629457161)

Last update: 2021/08/20 18:59