ACM Template

目录

[ACM Template 1](#_Toc142920639)

[数学 1](#_Toc142920640)

[欧拉函数 1](#_Toc142920641)

# 数学

## 实用数学结论

逆元存在的条件

在模p的群中 则称x为a的逆(inv).

逆元存在的条件是

鸽笼原理 Pigeonhole Principle：

13个人中至少2个人的生日在同一个月

找规律

通过通向表达式找规律，将数字用a，b等字母替代即可

中位数

中位数的性质：所有数与中位数的绝对差之和最小

## 整除分块

作用用的时间复杂度计算的问题

先得到处的的值然后计算这个值的右边界

1

…

…

n/5

图中橙色区域的值相同都等于而即得到的下标是整除能得到这个值的最大的位置，解释：直接用除法得到结果的余数最小且不会超过除数

1

…

…

//整除分块 复杂度O 根n

long long fenkuai(long long n) {

long long ans = 0;

for (long long l = 1, r; l <= n; l = r + 1) {

r = n / (n / l);

ans += (r - l + 1) \* (n / l);

}

return ans;

}

计算

if(n>m)swap(n,m);

for(int l=1;l<=n;l++){

int r=min(n/(n/l),m/(m/l);

int v1=n/l;

int v2=m/l;

ans+=(l-r)\*v1\*v2;

l=r;

}

## 欧拉函数

中与 互质的数的个数，被称为欧拉函数

// 1~N 中与 N 互质的数的个数，被称为欧拉函数

//时间复杂度为根号n

inline int euler\_one(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;

}

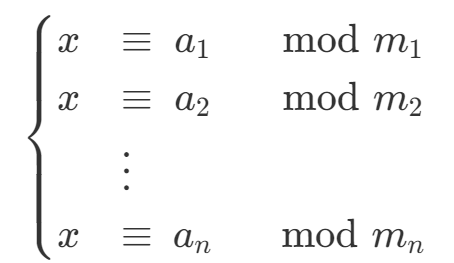
}

if (n > 1) ans = ans / n \* (n - 1); // n至多有一个比根号n大的质因子

return ans;

}

## 中国剩余定理：韩信点兵



1.若两两互质则有解

2. 在，下有唯一解为

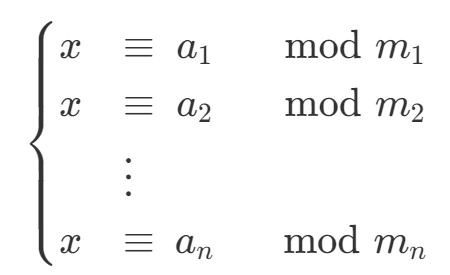
并设 是除了 以外的 个整数的乘积。

设 为 模 意义下的逆元。

理解性解释：

中国剩余定理主要用于解决这样的线性同余方程组，其中

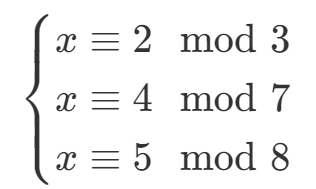
两两互质



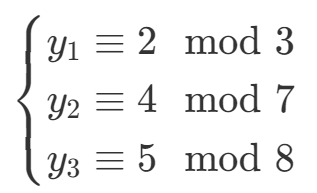
对于这样的方程组我们只需要求出 的一组特解，把这个特解加上 的最小公倍数的任意整数倍就可以得到所有解，所以我们就只需要考虑怎么构造出一组特解

例子

下面举个例子来帮助理解



直接构造不太行，但是对于每一个线性同余方程构造出一个特解是非常轻松的，所以我们先设第 个式子的解是 ，那么有



最终的 一定是 以各种方式组合起来，那么问题就是怎么组合才能得到

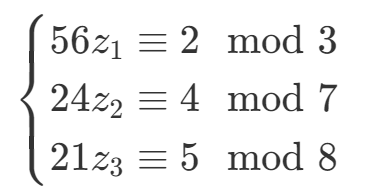
，如果 之间相乘就把问题变得更复杂了，所以我们把它们加起来，想办法凑出 如果 ，那么每一个 需要满足什么条件呢？因为 ，所以 一定是 3 的倍数，当 分别都是 3 的倍数时这一定也是满足条件的一个特解

所以我们不妨设 为3的倍数

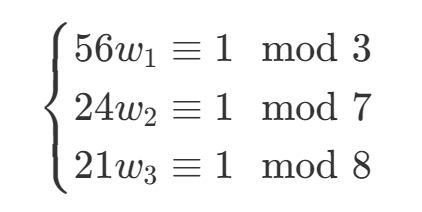
同理 为7的倍数，为8的倍数

那我们就可以把 写成 ，把 写成 ，把 写成

原方程就变成了这样：



然后就非常好求了，只需要求出



也就是 56,24,21 分别对于 3,7,8 的乘法逆元，然后在等式两边同时分别乘以 2,4,5 就可以得到 的一组特解,用扩展欧几里得算法求出 ，所以 于是

最后就得到 由于3,7,8的最小公倍数是168，我们将1229对于168取模就能得到 的最小正整数解

的通解就能表示为

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

#define int long long

const int N = 15;

int n, a[N], mods[N];

int exgcd(int a, int b, int& x, int& y) {

if (!b) {

x = 1, y = 0;

return a;

} else {

int gcd = exgcd(b, a % b, y, x);

y -= x \* (a / b); // 此处一定要先除后乘不然可能爆精度建议加括号

return gcd;

}

}

inline ll inv(ll x, ll p) // 求逆元

{

ll X, Y;

exgcd(x, p, X, Y);

return (X % p + p) % p;

}

inline ll crt() // CRT

{

ll prod = 1, ans = 0;

for (int i = 1; i <= n; ++i)

prod \*= mods[i];

ll tmp;

for (int i = 1; i <= n; ++i) {

tmp = prod / mods[i];

ans += (inv(tmp, mods[i]) \* a[i] \* tmp) % prod;

}

return ans % prod;

}

/\*

a[i]为每个方程式的余数

mods[i]为每个方程的模数

使用条件为所有mods[i]互质

\*/

signed example() {

cin >> n;

for (int i = 1; i <= n; ++i)

cin >> a[i] >> mods[i];

int ans = crt();

cout << ans;

}

## 数论基本知识

数论函数：定义域在整数上的函数

符号说明：(a,b)=c 相当于gcd(a,b)=c

积性函数：时

积性函数的性质： 则的全展开

积性函数f被所有处的值决定的

完全积性函数:不要求(a,b)=1即可f(ab)=f(a)f(b)完全积性函数也是积性函数

典型函数代表：

f(x)=1 I(x)=1

f(x)=x id(x)=x

f(x)=1 (x==1) =0(x!=1) e(x)=1 (x==1) =0(x!=1)

[x=1] 表示x==1时表达式=1，否则=0

中与 互质的数的个数，被称为欧拉函数

d(n) 因子个数

因子和

莫比乌斯函数

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | -1 | -1 | 0 | -1 | 1 | -1 | 0 | 0 | 1 |

// O(n)线性筛

// 其中p[i]表示i的最小素因子是多少，pr[tot]表示1~n范围内的质数

p[1]=1;

for(int i=2;i<=n;i++){

if(!p[i]) p[i]=i,pr[++tot]=I;

for(int j=1;j<=tot&&pr[j]\*i<=n;j++){

p[i\*pr[j]]=pr[j]; // 保证每个数只会被其最小的因子筛出

if(p[i]==pr[j])break; // 如果两者最小素因子相同就退出

}

}

2

3

5

7

9

1

1

+1

1

1

+1

1

1

2

1

break

// O(n)线性筛加求出指数

// 其中p[i]表示i的最小素因子是多少，pr[tot]表示1~n范围内的质数，pe[i]表示最小素因

// 子含指数一共是多少

p[1]=1;

for(int i=2;i<=n;i++){

if(!p[i]) p[i]=I,pe[i]=i,pr[++tot]=I;

for(int j=1;j<=tot&&pr[j]\*i<=n;j++){

p[i\*pr[j]]=pr[j]; // 保证每个数只会被其最小的因子筛出

if(p[i]==pr[j]){

pe[i\*pr[j]]=pe[i]\*pr[j];

break;

}else{

pe[i\*pr[j]]=pr[j];

}

}

}

线性筛筛出

i的情况：

1.pe[i] i为质数好求

2.i!=pe[i] i为合数 f[i]=f[i/pe[i]]\*f[pe[i]]

3.i=pe[i] i为质数的>=2次幂

// 线性筛筛出d sigma varphi mu O(n)

fd[1]=1;

for(int i=2;i<=n;i++){

if(i==pe[i])fd[i]=fd[i/p[i]]+1;

else fd[i]=fd[pe[i]]\*fd[i/pe[i]];

}

fsigma[1]=1;

for(int i=2;i<=n;i++){

if(i==pe[i]) fsigma [i]= fsigma [i/p[i]]+i;

else fsigma [i]= fsigma [pe[i]]\* fsigma [i/pe[i]];

}

fphi[1]=1;

for(int i=2;i<=n;i++){

if(i==pe[i])fphi[i]=i/p[i]\*(p[i]-1);

else fphi[i]=fphi[pe[i]]\*fphi[i/pe[i]];

}

fmu[1]=1;

for(int i=2;i<=n;i++){

if(i==pe[i]){

if(i==p[i])fmu[i]=-1;

else fmu[i]=0;

}else fmu[i]=fmu[pe[i]]\*fmu[i/pe[i]];

}

## 狄利克雷卷积

狄利克雷卷积是用来简化表示的

对于两个数论函数f，g => h=f\*g

狄利克雷卷积满足交换律

狄利克雷卷积满足结合律

重要：f和g 是积性函数 => f\*g 也是积性函数

## 莫比乌斯反演

莫比乌斯反演证明法1：

由证明积性函数只需要证质数的幂次得

证毕

莫比乌斯反演证明法2：

证毕

例题&模板：

给定一个函数 已知 求

在用上面的代码求出之间的的值之后用下面的代码求出g

// 通过求和式子求函数值写法 时间复杂度O(n)

for(int d1=1;d1<=n;d1++)

for(int d2=1;d1\*d2<=n;d2++){

g[d1\*d2]+=f[d1]\*mu[d2]

例题2：

问有多少对i,j满足且i，j互质

分析：

把d提出来

这样i，j就变得独立了

用分块整除优化。

if(n>m)sawp(n,m);

for(int l=1;l<=n;l++){  
 int v1=n/l;

int v2=m/l;

int r=min(n/v1,m/v2);

ans+=(summu[r]-summu[l])\*v1\*v2;

l=r;

}

小结：

形如 其中f是任意数论函数 (i,j)表示gcd(i,j)

例如例题2中f为e例题3中f为id

我们都可以搞出一个数论函数g使

于是

然后就把d提出来

例题3：

求

分析：

借助上面的小结，这题中f就相当于数论函数id

下面计算

首先因为其为积性函数，所以我们只用计算定义域为素数的幂次的位置即可

根据的定义只有d=1和d=p时项不等于0

时

时

于是

借助上面的结论

寄语：

刚刚的题目的做法不是偶然，对于上面小结中总结的题型，就是这种固定的做法，属于是学了就会不学就完全不会。前几年比较常考，但由于比较机械化，就没考的那么频繁了。 --杜瑜皓老师