基础数学

安玺儒

天津大学

2023 年 7 月 11 日

数学基础

1. Part 1. 预备知识
    - 数论基础
    - 线性代数
2. Part 2. 素数相关
    - 筛法
    - 欧拉函数
3. Part 3. 同余相关
    - 扩展欧几里得算法
    - 欧拉定理与扩展欧拉定理
    - 乘法逆元
    - 中国剩余定理与扩展中国剩余定理
4. Part 4. 组合数学
    - 排列与组合
    - 常用公式及原理
    - 卢卡斯定理与扩展卢卡斯定理

数学基础

Part 1. 预备知识

整除

tips: 数论基本只涉及到整数，因此所用字母除声明外都表示整数。

定义

设 $a, b \in \mathbb{Z}$，$b \neq 0$。
若 $\exists c \in \mathbb{Z}$，使得 $a = bc$，则称 b 整除 a，记作 $b \mid a$。
反之，则称 b 不能整除 a，记作 $b \nmid a$。

性质

- $a|b \iff -a|b \iff a|-b \iff |a|\,|\,|b|$
- $a|b \wedge b|c \Rightarrow a|c$
- $c|a \wedge c|b \Rightarrow \forall n, m \in \mathbb{Z}, c|(ma+nb)$
- $a|b \wedge a|c \Rightarrow a|(b \pm c)$
- $a|b \Rightarrow |a| \leq |b|\ (b \neq 0)$
- $a|b \wedge b|a \Rightarrow a = \pm b$
- $a|b \Rightarrow ma|mb\ (m \neq 0)$

Part 1. 预备知识

约数

定义

设 $a, b \in \mathbb{Z}$，若 $a|b$，则称 a 是 b 的约数 (因数，因子)，b 是 a 的倍数。

一些概念

显然约数 (因数)：对于 $b \neq 0$，±1、$\pm b$ 是 b 的显然约数。特殊地，当 $b = \pm 1$ 时，b 只有两个显然约数。
真约数 (因数)：对于 $b \neq 0$，b 除显然约数外的约数称为真约数。

性质

- 对于 $b \neq 0, b$ 的约数只有有限个。
- 对于 $b \neq 0$, 当 d 遍历 b 的全体约数时，$\frac{b}{d}$ 也遍历 b 的全体约数。
- 对于 $b > 0$, 当 d 遍历 b 的全体正约数时，$\frac{b}{d}$ 也遍历 b 的全体正约数。

Part 1. 预备知识

素数

概念

对于一个正整数，如果它有且仅有 1 和它自己两个约数，那么我们称这个数为素数 (Prime Number)。如果它有两个以上的约数，那么我们称这个数为合数 (Composite Number)。
注意：1 既不是素数也不是合数。

素数的性质

- 素数定理: 设 $x > 1$, 以 $\pi(x)$ 表示不超过 x 的素数的个数，当 $x \to \infty$ 时,$\pi(x) \to \frac{x}{ln(x)}$。
- 算术基本定理:$\forall n > 1$，可以将其表示为 $n = p_1 p_2 \ldots p_s$，其中 $p_j (1 \leq j \leq s)$ 为素数，并且在不计次序的情况下该表示唯一。将相同的素数合并有 $n = \prod\limits_{i=1}^{k} p_i^{\alpha_i}$，其中，$p_1 < p_2 < \ldots < p_k$。该式称为正整数 a 的标准素因数分解式。

Part 1. 预备知识

同余

概念

设 n 是给定的正整数，若整数 a、b 满足 $n \mid (a-b)$，则称 a 和 b 模 n 同余，记作 $a \equiv b(mod\ n)$。反之，则称 a 与 b 模 n 不同余，记作 $a \not\equiv b(mod\ n)$。

同余的性质

- 反身性：$a \equiv a(mod\ n)$。
- 对称性：$a \equiv b(mod\ n) \iff b \equiv a(mod\ n)$。
- 传递性：$a \equiv b(mod\ n) \wedge b \equiv c(mod\ n) \Rightarrow a \equiv c(mod\ n)$。
- 可加性：$a \equiv b(mod\ n) \wedge c \equiv d(mod\ n) \Rightarrow a \pm c \equiv b \pm d(mod\ n)$。
- 可乘性：$a \equiv b(mod\ n) \wedge c \equiv d(mod\ n) \Rightarrow ac \equiv bd(mod\ n)$。

Part 1. 预备知识

公约数与公倍数

概念

- 公约数 (Common Divisor): 如果一个整数同时是几个正整数的约数，称这个整数为它们的公约数. 特别地，对任意若干整数，± 1 总是它们的公约数。
- 最大公约数 (Greatest Common Divisor): 一组整数公约数中最大的称为最大公约数。记作 $(a_1, a_2, \dots, a_n)$。
- 公倍数 (Common Multiple) 如果一个整数同时是几个整数的倍数，称这个整数为它们的公倍数。
- 最小公倍数 (Least Common Multiple): 一组整数公倍数中最小的正整数称为最小公倍数。记作 $[a_1, a_2, \dots, a_n]$

一般地，我们只研究两个正整数的最小公倍数和最大公约数。在算法竞赛中，为方便描述，将 (a, b) 写为 $gcd(a, b)$，将 $[a, b]$ 写为 $lcm(a, b)$。

Part 1. 预备知识

性质

一些性质

- $gcd(a,b) = gcd(b,a)$
- $gcd(a,b) = gcd(a, b \pm ka)\ (k \neq 0)$
- $gcd(a,a) = a$
- $gcd(a,b) = gcd(a\ \ mod\ \ b, b)$
- $gcd(a,b) \times lcm(a,b) = ab$
- $gcd(a,1) = 1$

由性质 2, 3，我们可以将 $gcd(a,0)$ 定义为 a。

Part 1. 预备知识

互素

定义

- 两个整数互素的定义：对于 $a, b \in \mathbb{Z}$, 如果 $gcd(a, b) = 1$，则称这两个数互素 (互质，既约)。
- 多个整数互素的定义：对于 $a_1, a_2, ..., a_n \in \mathbb{Z}$，如果 $gcd(a_1, a_2, ..., a_n) = 1$，则称这 n 个数互素 (互质，既约)。
- 一般只研究两个数之间的互素。
- 注意：多个数互素，不一定两两互素，例如 $6, 10, 15$，这三个数互素，但任意两个都不互素。

Part 1. 预备知识

例题

luogu P1372 又是毕业季 I

题目大意：给定 n, k，在 $1 - n$ 中取 k 个数，使这 k 个数的公约数最大，其中 $n, k \leq 10^9$。

Part 1. 预备知识

例题

luogu P1372 又是毕业季 I

题目大意：给定 n, k，在 $1-n$ 中取 k 个数，使这 k 个数的公约数最大，其中 $n, k \leq 10^9$。

Solution

设取的 k 个数分别为 $a_1, a_2, ..., a_k, gcd(a_1, a_2, ..., a_k) = m$。
由贪心的思想，想让 gcd 尽可能得大，可以取 $a_1 = m, a_2 = 2m, ..., a_k = km$。
在 $1-n$ 中进行取数，故有 $km \leq n$，因此 $gcd_{max} = m = \lfloor \frac{n}{k} \rfloor$。

Part 1. 预备知识

求解最大公约数

算法竞赛中最大公约数应用较多，一般用如下两个算法求解。

辗转相除法 (Euclid's Algorithm)

前面提到了一个性质 $gcd(a, b) = gcd(b, a \ mod \ b)$，因此我们利用这个性质可以逐步递归求解两个数的最大公约数。时间复杂度 $O(log_b)$。

```
int gcd(int a, int b){
    if(b==0)return a;
    else return gcd(b,a%b);
```

压行：

```
int gcd(int a, int b){
    return b?gcd(b,a%b):a;
}
```

tips: $C++$ 中 $algorithm$ 库里有 $\_\_gcd$ 函数，可以直接使用。

Part 1. 预备知识

求解最大公约数

更相减损术

大整数取模的时间复杂度较高，而加减法时间复杂度较低。针对大整数，我们可以用加减代替乘除求出最大公约数。即利用性质 $gcd(a,b)=gcd(a,b\pm ka)\ (k\neq 0)$。
但如果 $a>>b$，更相减损术的时间复杂度会达到 $O(n)$ 的最坏情况，因此我们考虑优化。

Part 1. 预备知识

求解最大公约数

Stein's Algorithm

若 $2 \mid a, 2 \mid b$, 则 $gcd(a,b) = 2gcd(\frac{a}{2}, \frac{b}{2})$;
若 $2 \mid a, 2 \nmid b$, 则 $gcd(a,b) = gcd(\frac{a}{2}, b)$;
若 $2 \nmid a, 2 \mid b$, 则 $gcd(a,b) = gcd(a, \frac{b}{2})$;
若 $2 \nmid a, 2 \nmid b$, 则 $gcd(a,b) = gcd(a, a-b)$;
时间复杂度 $O(\log n)$。

代码实现

```
int stein(int a,int b){
    if(a<b)a^=b,b^=a,a^=b;
    if(b==0)return a;
    if((!(a&1))&&(!(b&1)))return stein(a>>1,b>>1)<<1;
    else if((a&1)&&(!(b&1)))return stein(a,b>>1);
    else if((!(a&1))&&(b&1))return stein(a>>1,b);
    else return stein(a-b,b);
}
```

Part 1. 预备知识

例题

luogu P2090 数字对

题目大意：对于一个数字对 (a, b)，我们可以通过一次操作将其变为新数字对 $(a+b, b)$ 或 $(a, a+b)$。
给定一正整数 n，问最少需要多少次操作可将数字对 $(1, 1)$ 变为一个数字对，该数字对至少有一个数字为 n。
其中 $n \leq 10^6$。

Part 1. 预备知识
例题

Solution

这道题用到的是递归思想，和数论没什么太大的关系，把这道题放这是为了强化大家对辗转相除法过程的理解。
不难发现，变换数字对的过程是辗转相除法的一个逆过程，那么我们可以在求 gcd 的同时维护变换次数，从而得到答案。
两个性质：1. 数字对 $(1,1)$ 变换为 (a,b) 和 (b,a) 的操作次数是相同的。
2. 数字对中首次出现数字 n， n 一定是两个数中最小的。
因此只要遍历 1 到 $n-1$，找出与 n 辗转相除的递归层数最小的即可。

Part 1. 预备知识

例题

代码实现

```cpp
#include<bits/stdc++.h>
#define inf int(1e8)
using namespace std;
int gcdd(int a, int b){
    if(!b)return inf;
    if(b==1)return a-1;
    return gcdd(b,a%b)+a/b;
}
int n,now=inf;
int main(){
    cin>>n;
    for(int i=1;i<=(n+1)/2;i++)now=min(now,gcdd(n,i));
    cout<<now<<endl;
}
```

数学基础

Part 1. 预备知识

线性代数

简介

线性代数是数学的一个分支，它的研究对象是向量，向量空间 (或称线性空间)，线性变换和有限维的线性方程组。在算法竞赛中，线性代数的知识可以直接用来解决问题，也可以用于优化算法、数据结构等。例如今天要介绍的：用矩阵优化递推。

由于各位应该都学过线性代数，因此对于线性代数知识就不一一赘述了，只介绍与算法竞赛相关的知识点。

Part 1. 预备知识

矩阵乘法

引入

矩阵乘法的引入来源于线性方程组。

例如，将线性方程组 $\begin{cases} x_1 + 2x_2 + 3x_3 = 25 \\ 4x_4 + 5x_5 + 6x_6 = 26 \\ 7x_7 + 8x_8 + 9x_9 = 27 \end{cases}$

转化为矩阵乘法的形式：$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 25 \\ 26 \\ 27 \end{bmatrix}$。简记为 $Ax = B$

应用

矩阵乘法的应用之一是加速递推，原理是将递推式化为矩阵的形式，如 $Ax = B$。将 n 次递推化为做 n 次矩阵乘法，如 $(((Ax)x)\ldots)x$，由于矩阵乘法的结合律，该式可化为 Ax^n，再用快速幂计算 x^n，可将递推的时间复杂度由 $O(n)$ 降为 $O(log_n)$。

Part 1. 预备知识

矩阵乘法的应用

luoguP1962 斐波那契数列

斐波那契数列 (The Fibonacci sequence，OEIS A000045)：
$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$。
该数列的前几项为:$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 87 \ldots$
求 $F_n \, mod \, 10^9 + 7$，其中 $1 \leq n \leq 2^{63}$。

朴素做法

循环遍历 n 次，从 F_1, F_2 一直递推到 F_n，时间复杂度 $O(n)$。
由于 n 为 2^{63}，时间消耗过大，考虑应用矩阵乘法优化。

Part 1. 预备知识

矩阵乘法的应用

Solution

构造 A 矩阵为 $A_n = \begin{bmatrix} F_n & F_{n-1} \end{bmatrix}$，递推矩阵 x 为 $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$。

则一次矩阵乘法后的结果为

$A_n x = \begin{bmatrix} F_n & F_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} F_{n+1} & F_n \end{bmatrix} = A_{n+1}$。

因此我们通过矩阵乘法实现了递推，所以我们只需求解 $A_1 x^{n-1}$，最后输出矩阵的第一项即可。

Part 1. 预备知识

矩阵乘法的应用

代码实现

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod=ll(1e9+7);
struct node{
    ll sqr[5][5];
}a;
node operator*(node a, node b){
    node c;
    c.sqr[1][1]=(a.sqr[1][1]*b.sqr[1][1]%mod+a.sqr[1][2]*b.sqr[2][1]%mod)%mod;
    c.sqr[1][2]=(a.sqr[1][1]*b.sqr[1][2]%mod+a.sqr[1][2]*b.sqr[2][2]%mod)%mod;
    c.sqr[2][1]=(a.sqr[2][1]*b.sqr[1][1]%mod+a.sqr[2][2]*b.sqr[2][1]%mod)%mod;
    c.sqr[2][2]=(a.sqr[2][1]*b.sqr[1][2]%mod+a.sqr[2][2]*b.sqr[2][2]%mod)%mod;
    return c;
}
ll n;
void quickpow(node &x, ll y){
    node rec;
    rec.sqr[1][1]=rec.sqr[2][2]=1,rec.sqr[1][2]=rec.sqr[2][1]=0;
    while(y){
        if(y&1)rec=rec*x;
        x=x*x,y>>=1;
    }
    x=rec;
}
int main(){
    scanf("%lld",&n);
    if(n==0)return puts("0");
    a.sqr[1][1]=a.sqr[1][2]=a.sqr[2][1]=1,a.sqr[2][2]=0;
    quickpow(a,n-1);
    printf("%lld\n",a.sqr[1][1]);
}
```

Part 1. 预备知识

矩阵的其他应用

加速其他形式的递推

如 $f_n = 7f_{n-1} + 6f_{n-2} + 5n + 4 \times 3^n$。

高斯消元

给定一个一元线性方程组，对其求解，留作思考题。

数学基础

Part 2. 素数相关

判断素数

方法

对于 $n > 1$，如果 2 到 $n-1$ 每个数都不是 n 的因数，那么 n 就是质数了。因此可以考虑遍历 2 到 $n-1$ 一一判断。
时间复杂度 $O(n)$。

优化

考虑对进行优化，上面提到过约数的一个性质：对于 $b > 0$，当 d 遍历 b 的全体正约数时，$\frac{b}{d}$ 也遍历 b 的全体正约数。
因此我们只需遍历 2 到 $\lfloor\sqrt{n}\rfloor$ 的数。

Part 2. 素数相关

埃氏筛

引题

求 $1-n$ 内的素数。$n \leq 10^6$

埃氏筛

- 原理：素数的倍数不是素数
- 基本思路：每次找到一个素数时，将素数的倍数全部筛去。
- 时间复杂度为 $O(nlog(log(n)))$

```
bool vis[N];//vis[i]为真表示i不是素数
int prime[N],cnt=0;//prime数组存的是素数 cnt为当前素数个数
for(int i=2;i<=n;i++){
    if(!vis[i]){//没被筛去说明是素数
        prime[++cnt]=i;//存下新的素数
        for(int j=1;j*i<=n;j++)
            vis[i*j]=true;//筛去素数的倍数
    }
}
```

Part 2. 素数相关

线性筛

线性筛

顾名思义，线性筛会将时间复杂度降低到线性，即 $O(n)$。

- 基本思路：埃氏筛中，一个数会被筛去多次，如 15 会被 3 和 5 各筛去了一次。如果我们让一个数只会被其素数约数中最小的筛去一次，可以大大节省时间。
- 实现方法：假设目前筛到 x，想将 x 的倍数筛去。

  将 x 用标准素因数分解式表示为 $x = \prod_{i=1}^{k} p_i^{\alpha_i}$，$p_1 < p_2 < \ldots < p_k$。

  我们考虑让 x 的倍数 kx 只被其素因数最小的素数筛去。

  因此 k 要满足:k 为素数且 $k \leq p_1$，故只需将 x 的 $2, 3, 5, \ldots, p_1$ 倍筛去即可。

Part 2. 素数相关

线性筛

代码实现

```
bool vis[N];//vis[i]为真表示i不是素数
int prime[N],cnt=0;//prime数组存的是素数 cnt为当前素数个数
for(int i=2;i<=n;i++){//线性筛
    if(!vis[i])prime[++cnt]=i;//没被筛到说明是素数
    for(int j=1;j<=cnt&&i*prime[j]<=n;j++){//注意该循环遍历的是素数
        vis[i*prime[j]]=true;//筛去合数
        if(i%prime[j]==0)break;//如果i是j最小的素因子跳出循环
    }
}
```

数学基础

Part 2. 素数相关

欧拉函数

定义

欧拉函数，即 $\phi(x)$，它表示从 $1-n$ 中和它互质的数的个数。
例如 $\phi(1)=1, \phi(2)=1, \phi(6)=2$。

性质

- 对于素数 n,$\phi(n)=n-1$。
- 若 $n=p^k$，其中 p 时质数，那么 $\phi(n)=p^{k-1}(p-1)$。
- 积性:$\forall n, m \in \mathbb{N}+ \wedge (n,m)=1 \Rightarrow \phi(nm)=\phi(n)\phi(m)$。
  特别地，当 $2 \nmid n$ 时，$\phi(2n)=\phi(n)$。
- $\forall n, m \in \mathbb{N}+ \wedge n \mid m \Rightarrow \phi(nm)=n\phi(m)$。

Part 2. 素数相关

欧拉函数的求解

单个数的欧拉函数

$\forall x \in \mathbb{N}+$，由算术基本定理，$x = \prod_{i=1}^{n} p_i^{\alpha_i}$ 那么 $\phi(x) = x\prod_{i=1}^{n}(1-\frac{1}{p_i})$。

证明

利用上述性质 2, 3。

证明:$\phi(x) = \phi(\prod_{i=1}^{n} p_i^{\alpha_i}) = \prod_{i=1}^{n}\phi(p_i^{\alpha_i}) = \prod_{i=1}^{n} p_i^{\alpha_i-1}(p_i-1) =$
$\prod_{i=1}^{n} p_i^{\alpha_i}(1-\frac{1}{p_i}) = \prod_{i=1}^{n} p_i^{\alpha_i}\prod_{i=1}^{n}(1-\frac{1}{p_i}) = x\prod_{i=1}^{n}(1-\frac{1}{p_i})$。

代码实现

进行质因数分解的同时维护即可。

Part 2. 素数相关

求多个数的欧拉函数

引题

求 $1-n$ 所有数的欧拉函数。

朴素算法

对每个数直接使用用欧拉函数求解公式，时间复杂度为 $O(n\sqrt{n})$。

优化

考虑如何进行优化。
上面已经提到如下性质：

- 对于素数 $p, \phi(p)=p-1$。
- $\forall n,m \in \mathbb{N}+ \wedge (n,m)=1 \Rightarrow \phi(nm)=\phi(n)\phi(m)$。
- $\forall n,m \in \mathbb{N}+ \wedge n \mid m \Rightarrow \phi(nm)=n\phi(m)$ 。

Part 2. 素数相关

求多个数的欧拉函数

筛法求解

考虑如何进行优化。
上面已经提到如下性质：

- 对于素数 $p, \phi(p) = p - 1$
- $\forall n, m \in \mathbb{N}+ \wedge (n, m) = 1 \Rightarrow \phi(nm) = \phi(n)\phi(m)$
- $\forall n, m \in \mathbb{N}+ \wedge n \mid m \Rightarrow \phi(nm) = n\phi(m)$

回想到线性筛的过程：
如果 x 为质数，直接更新当前 x 处的信息。
对于每个数 x，都用已有的信息更新 x 的倍数处的信息。
再结合上述三个性质，我们就可以通过筛法的过程维护每个数的欧拉函数。

Part 2. 素数相关

代码实现

代码实现

```cpp
bool vis[N];//vis[i]为真表示i不是素数
int cnt,prime[N],phi[N];//prime数组存的是素数 cnt为当前素数个数
phi[1]=1;//初始化
for(int i=2;i<=n;i++){
    if(!vis[i])phi[i]=i-1,prime[++cnt]=i;//性质1
    for(int j=1;j<=cnt&&i*prime[j]<=n;j++){
        vis[i*prime[j]]=true;
        if(i%prime[j]==0){//性质3
            phi[i*prime[j]]=phi[i]*prime[j];
            break;
        }
        else
            phi[i*prime[j]]=phi[i]*phi[prime[j]];//性质2
    }
}
```

Part 2. 素数相关

例题

简单题

题目大意：给定 n，求 $1 \leq x, y \leq n$ 且 $gcd(x, y) = 1$ 的数对个数。

Part 2. 素数相关

例题

简单题

题目大意：给定 n, 求 $1 \leq x, y \leq n$ 且 $gcd(x, y) = 1$ 的数对个数。

Solultion

用式子写出来就是: $\sum_{i=1}^{n}\sum_{j=1}^{n}(gcd(i,j) = 1) = \sum_{i=1}^{n}\phi(i)$,
所以只需要求出欧拉函数的前缀和即可。

Part 2. 素数相关

例题

Solution

```cpp
#include<bits/stdc++.h>
using namespace std;
#define N 1000000
bool vis[N];
int n,cnt,prime[N],phi[N];
long long sum=0;
int main(){
    cin>>n;
    phi[1]=1;
    for(int i=2;i<=n;i++){
        if(!vis[i])phi[i]=i-1,prime[++cnt]=i;
        for(int j=1;j<=cnt&&i*prime[j]<=n;j++){
            vis[i*prime[j]]=true;
            if(i%prime[j]==0){
                phi[i*prime[j]]=phi[i]*prime[j];
                break;
            }
            else
                phi[i*prime[j]]=phi[i]*phi[prime[j]];
        }
    }
    for(int i=1;i<=n;i++)sum+=phi[i];
    cout<<sum<<endl;
}
```

数学基础

Part 3. 同余相关

贝祖定理

引题

给定 a, b, c, 求使得 $ax + by = c$ 成立的最小正整数解 x, y。

无解条件

首先考虑是否有解，观察方程左侧，有 $gcd(a, b) | ax + by$,
因此当 $gcd(a, b) \nmid c$ 时无解。

有解条件

贝祖 (裴蜀) 定理 (Bézout's Lemma)：存在整数 x, y, 使得 $ax + by = gcd(a, b)$ 成立。
其中 $ax + by = gcd(a, b)$ 称为贝祖等式 (Bézout's Identity)。
一个重要推论：$gcd(a, b) = 1 \iff \exists x, y \in \mathbb{Z}$, 使得 $ax + by = 1$。
因此当 $gcd(a, b) \mid c$ 时方程有解。

Part 3. 同余相关

扩展欧几里得算法

下面介绍贝祖等式的一种解法，即扩展欧几里得算法 (Extended Euclid's Algorithm)。

扩展欧几里得算法 (Extended Euclid's Algorithm)

该算法运用的思想是从特殊情况到一般情况的一种递归思想。
当 $a = gcd(a, b), b = 0$ 时，方程化为 $gcd(a, b)x = gcd(a, b)$，
一组解为 $x = 1, y = m, m \in \mathbb{Z}$。
由于已经知道末状态的通解，我们只需找出相邻两个方程通解之间的联系，即可递推求出贝祖等式的通解。
$bx_1 + (a \ mod \ b)y_1 = bx_1 + (a - \lfloor \frac{a}{b} \rfloor b)y_1$
$= ay_1 + b(x_1 - \lfloor \frac{a}{b} \rfloor y_1) = gcd(a, b) = ax_2 + by_2$。
因此我们得到了一组递归式，即 $x_2 = y_1, y_2 = x_1 - \lfloor \frac{a}{b} \rfloor y_1$。

Part 3. 同余相关

扩展欧几里得的代码实现

代码实现

数学推导过程较为繁琐，因为应用递归的思想，代码实现会更简洁。方程的通解是随着 (a, b) 一步一步递归到 $(gcd(a, b), 0)$ 得到的，因此我们在递归求 gcd 时维护通解即可求得结果。

```
void extended_gcd(int a, int b, int &x, int &y){
    if(b==0){x=1,y=0;return;}
    extended_gcd(b,a%b,x,y);
    int t=y;
    y=x-(a/b)*y,x=t;
}
```

Part 3. 同余相关

例题

NOIP2012 提高组　同余方程

题目大意：求关于 x 的同余方程 $ax \equiv 1 \pmod{b}$ 的最小正整数解 (保证方程有解)。

Part 3. 同余相关

例题

Solution

不难发现可以转化为 $ax + by = 1$，运用 *exgcd* 即可求得最小正整数解。

```
#include<bits/stdc++.h>
using namespace std;
int t,a,b,x,y;
void exgcd(int a, int b, int &x, int &y){
    if(b==0){x=0,y=1;return;}
    exgcd(b,a%b,x,y);
    t=x,x=y,y=t-a/b*y;
}
int main(){
    cin>>a>>b;
    exgcd(a,b,x,y);
    while(x<=0){
        x+=b;
    }
    cout<<x<<endl;
}
```

数学基础

Part 3. 同余相关

欧拉定理

定义

欧拉定理 (Euler's Theorem): 若 $gcd(a, m) = 1$，则 $a^{\phi(m)} \equiv 1(mod\ m)$。

证明

欧拉定理的证明涉及到剩余类、剩余系、完系、缩系的概念。
(1) 剩余类：对于正整数 n，把所有整数根据模 n 的余数 $r \in [0, n-1]$ 分为 n 类，这样的一类数所构成的一个集合称为模 n 的剩余类。
(2) 剩余系：对于正整数 n，任选 x 个不同的模 n 的剩余类，每个剩余类中各取出一个元素，我们称这 x 个数构成的集合为模 n 的剩余系。
(3) 完系：对于正整数 n，在其 n 个不同的模 n 的剩余类中各取出一个元素，我们称这 n 个数构成的集合为模 n 的完全剩余系 (完系)。
(4) 缩系：对于正整数 n，有 $\phi(n)$ 个模 n 的余数 r 使得 $gcd(r, n) = 1$，从这 $\phi(n)$ 个剩余系中各取出一个元素，将这 $\phi(n)$ 个数构成一个新的集合，称其为模 n 的简化剩余系 (缩系)。

Part 3. 同余相关

扩展欧拉定理

引题

给定 a, n, m 求 $a^n \ mod\ k$

朴素算法

用循环一个一个累乘在 $O(n)$ 的时间内求出答案。
如果 n 的范围是 $n \leq 10^9$，这样的方法貌似行不通。

快速幂

快速幂本质上利用了分治的思想。考虑到 $a^{2n} = (a^n)^2$ 所以我们可以两两配对地乘。
当 n 是奇数时，那么有 $a^n = a * a^{n-1}$
当 n 是偶数时，那么有 $a^n = a^{\frac{n}{2}} * a^{\frac{n}{2}}$
如果 n 再大呢，比如 $n \leq 10^{200000}$，快速幂也不能解决。

Part 3. 同余相关

扩展欧拉定理

定义

$$a^b \equiv \begin{cases} a^{b \ mod \ \phi(m)} & , gcd(a,m)=1 \\ a^{(b \ mod \ \phi(m))+\phi(m)} & , gcd(a,m)\neq 1, b \geq \phi(m) \\ a^b & , gcd(a,m)\neq 1, b < \phi(m) \end{cases} \quad (mod \ m)$$

证明

$$a = p \Longrightarrow a = p^k \Longrightarrow a = \prod p_i^{k_i}$$

应用

欧拉降幂

Part 3. 同余相关

例题

luogu P5091 【模板】扩展欧拉定理

给你三个整数:a, m, b，你需要求 $a^b \mod m$，其中 $1 \leq a \leq 10^9, 1 \leq b \leq 10^{20000000}, 1 \leq m \leq 10^8$。

Solution

应用扩展欧拉定理，进行欧拉降幂即可。

```cpp
int main(){
    cin>>a>>m;nm=m,a%=m;
    for(int i=2;i*i<=nm;i++){
        if(nm%i==0){
            phi*=(i-1),nm/=i;
            while(nm%i==0){phi*=i,nm/=i;}
        }
    }if(nm>1)phi*=(nm-1);
    char ch=getchar();
    while(ch<'0'||ch>'9')ch=getchar();
    while(ch>='0'&&ch<='9'){
        b=(b<<3)+(b<<1)+(ch&15);ch=getchar();
        if(b>=phi)flag=1,b%=phi;
    }
    if(flag)b+=phi;
    cout<<qpow(a,b)<<endl;
}
```

数学基础

Part 3. 同余相关

乘法逆元

引题

给定 n, m, p, 求 $\frac{n}{m}$ *mod* p, 其中 $m \not\equiv 0 \pmod{p}$, 其中 $(gcd(m, p) = 1)$。
想要求解这个问题，便需要引入逆元的概念。

定义

对于 $a, p \in \mathbb{N}+ \wedge gcd(x, p) = 1$, 如果 $\exists x$ 使得 $ax \equiv 1 \pmod{p}$，则称 x 为 a 模 p 意义下的乘法逆元，记作 $x \equiv a^{-1} \pmod{p}$。
当 $gcd(x, p) \neq 1$ 时逆元不存在。
下面介绍三种求逆元的方法。

Part 3. 同余相关

求逆元的三种方法

exgcd 求逆元

观察逆元的性质，$ax \equiv 1 \pmod p$, 和上面那道例题同余方程一样。
因此得到了求逆元的第一种方法：*exgcd*。
具体思路和实现刚刚已经讲过，不在这里赘述。
适用范围：求一个或几个数的逆元时可以用得到，当数过多时不适用。

Part 3. 同余相关

求逆元的三种方法

费马小定理

Fermat′s Little Theorem: 当 p 为质数时，$a^{p-1} \equiv 1 \pmod{p}$。

Part 3. 同余相关

求逆元的三种方法

费马小定理

Fermat's Little Theorem: 当 p 为质数时，$a^{p-1} \equiv 1 \pmod{p}$。

证明

费马小定理的证明涉及到刚刚讲的欧拉定理。
Euler Theorem： $a^{\phi(p)} \equiv 1 \pmod{p}$, 其中 $(a, p) = 1$。
特别地，当 p 为质数时，有 $\phi(p) = p - 1$，因此 $a^{p-1} \equiv 1 \pmod{p}$。

快速幂求逆元

当模数为质数时，有 $a^{p-2} \times a \equiv 1 \pmod{p}$，因此 a^{p-2} 为 a 的逆元。
适用范围：模数是质数，求一个数或几个数的逆元时可以用到，当数过多或模数不是质数时不适用。
适用范围看似很小，但由于题目中的模数多为大质数，故应用广泛。

Part 3. 同余相关

求逆元的三种方法

线性求逆元

当我们一次要求多个数的逆元，用费马小定理和 *exgcd* 就过于消耗时间，我们考虑如何在线性的时间内求解多个数的逆元。
模数为 p 时，对于数 $i>1$，我们让 p 对 i 做带余除法:
$p=ki+r$, 其中 $k=\lfloor \frac{p}{i} \rfloor, r\in[0,i)$ 且 $r=p\ mod\ i$。

等式两侧模 p，有:$ki+r\equiv 0 \pmod p$,
等式两侧乘 $i^{-1}r^{-1}$, 有 $kr^{-1}+i^{-1}\equiv 0 \pmod p$,

因此 $i^{-1}\equiv -kr^{-1} \pmod p$, 代入有 $i^{-1}\equiv -\lfloor \frac{p}{i} \rfloor (p\ mod\ i)^{-1}$。

注意到 $p\ mod\ i<i$，因此我们可以递推出 i 的逆元，
递推公式 $inv[i]=((p-p/i)inv[p\%i])\%p$。

```
int inv[N],mod;inv[1]=1;
for(int i=2;i<mod;i++)inv[i]=(mod-mod/i)*inv[mod%i]%mod;
```

Part 3. 同余相关

例题

luogu P2613 有理数取余

题目大意：给定 a, b，求 $c = \frac{a}{b}$ mod 19260817，如果无解输出 *Angry!*。其中 $a, b \le 10^{10001}$。

Part 3. 同余相关

例题

luogu P2613 有理数取余

题目大意：给定 a, b，求 $c = \frac{a}{b}$ mod 19260817，如果无解输出 $Angry!$。其中 $a, b \leq 10^{10001}$。

Solution

a, b 的数据范围非常大，所以我们一位一位输入，边输入边取模，最后用 a 与 b 对 19260817 取得的模数进行求解。
当 $b \equiv 0 \pmod{19260817}$ 时无解，其余情况输出 a 与 b 的逆元乘积对 19260817 取模即可。由于 19260817 是质数，所以我们这里可以用费马小定理求逆元。

Part 3. 同余相关

例题

Solution

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mod=19260817;
ll quickpow(ll a, ll b){
    ll ret=1;
    while(b){
        if(b&1)(ret*=a)%=mod;
        (a*=a)%=mod,b>>=1;
    }
    return ret;
}
ll n,m;
ll getint(){
    char chr=getchar();ll x=0;
    while(chr<'0'||chr>'9')chr=getchar();
    while(chr>='0'&&chr<='9'){x=(x*10)%mod;chr=getchar();}
    return x;
}
int main(){
    n=getint(),m=getint();
    if(m==0)return printf("Angry!"),0;
    printf("%lld\n",n*quickpow(m,mod-2)%mod);
}
```

数学基础

1. Part 1. 预备知识
   - 数论基础
   - 线性代数
2. Part 2. 素数相关
   - 筛法
   - 欧拉函数
3. Part 3. 同余相关
   - 扩展欧几里得算法
   - 欧拉定理与扩展欧拉定理
   - 乘法逆元
   - 中国剩余定理与扩展中国剩余定理
4. Part 4. 组合数学
   - 排列与组合
   - 常用公式及原理
   - 卢卡斯定理与扩展卢卡斯定理

Part 3. 同余相关
中国剩余定理

引题

有物不知其数，三三数之剩二，五五数之剩三，七七数之剩二。问物几何? ——《孙子算经》
即求满足以下条件的整数：除以 3 余 2，除以 5 余 3，除以 7 余 2。

概念

中国剩余定理 (Chinese Remainder Theorem): 也称孙子定理，可求解如下形式的一元线性同余方程组，其中 $(m_1, m_2, ..., m_k) = 1$,

$$\begin{cases} x \equiv a_1 (mod\ m_1) \\ x \equiv a_2 (mod\ m_2) \\ ... \\ x \equiv a_k (mod\ m_k) \end{cases}$$

上面的问题就是一元线性同余方程组的一个实例。

Part 3. 同余相关

中国剩余定理

求解过程

求解的主要思想是构造法，构造出一个解。取 $M = \prod_{i=1}^{k} m_i$；
对于第 i 个方程，设 $p_i = \frac{M}{m_i}$，求出 p_i 在模 m_i 下的逆元为 p_i^{-1}；
则有一个解 $x_0 = \sum\limits_{i=1}^{k} a_i p_i p_i^{-1}$。

Part 3. 同余相关

中国剩余定理

代码实现

```
typedef long long ll;
ll a[N],b[N],t[N],k;
//b数组为模数，t数组为逆元
void ex_gcd(ll n, ll m, ll &x, ll &y){
    if(!m){x=1,y=0;return;}
    ex_gcd(m,n%m,y,x);y-=(n/m)*x;
}
ll CRT(){
    ll ans=0,lcm=1,x,y;
    for(int i=1;i<=k;i++)lcm*=b[i];
    for(int i=1;i<=k;i++){
        t[i]=lcm/b[i];
        ex_gcd(t[i],b[i],x,y);
        x=(x%b[i]+b[i])%b[i];
        ans=(ans+t[i]*a[i]%lcm*x%lcm)%lcm;
    }
    return (ans+lcm)%lcm;
}
```

Part 3. 同余相关

例题

TJOI2009 猜数字

题目大意：求最小的 $n \in \mathbb{N}+$, 使得 $\begin{cases} m_1 \mid (n-a_1) \\ m_2 \mid (n-a_2) \\ \dots \\ m_k \mid (n-a_k) \end{cases}$,

其中 $1 \le k \le 10, |a_i| \le 10^9, 1 \le b_i \le 6 \times 10^3, \prod_{i=1}^{k} b_i \le 10^{18}$。

Part 3. 同余相关

例题

Solution

$$\begin{cases} m_1 \mid (n-a_1) \\ m_2 \mid (n-a_2) \\ \dots \\ m_k \mid (n-a_k) \end{cases} \text{可化为} \begin{cases} n \equiv a_1 (mod\ m_1) \\ n \equiv a_2 (mod\ m_2) \\ \dots \\ n \equiv a_k (mod\ m_k) \end{cases},$$

用中国剩余定理解决即可。

注意：题目数据过大，直接乘法会爆 *long long*，要应用快速乘，具体做法同快速幂，细节看代码。

Part 3. 同余相关

例题

Solution

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
inline ll quickpow(ll x, ll y, ll mod){
    ll ret=1;
    while(y){
        if(y&1)(ret*=x)%=mod;
        (x*=x)%=mod,y>>=1;
    }
    return ret;
}
inline ll quickmul(ll x, ll y, ll mod){
    if(x<0)x+=mod;if(y<0)y+=mod;
    if(y>x)swap(x,y);
    ll ret=0;
    while(y){
        if(y&1)(ret+=x)%=mod;
        (x+=x)%=mod,y>>=1;
    }
    return ret;
}
```

```cpp
ll k,a[20],b[20],t[20];
void ex_gcd(ll n, ll m, ll &x, ll &y){
    if(!m){x=1,y=0;return;}
    ex_gcd(m,n%m,y,x);y-=(n/m)*x;
}
ll CRT(){
    ll ans=0,lcm=1,x,y;
    for(int i=1;i<=k;i++)lcm*=b[i];
    for(int i=1;i<=k;i++){
        t[i]=lcm/b[i];
        ex_gcd(t[i],b[i],x,y);
        x=(x%b[i]+b[i])%b[i];
        ans=(ans+quickmul(quickmul(t[i],a[i],lcm),x,lcm))%lcm;
    }
    return (ans+lcm)%lcm;
}
int main(){
    scanf("%lld",&k);
    for(int i=1;i<=k;i++)scanf("%lld",&a[i]);
    for(int i=1;i<=k;i++)scanf("%lld",&b[i]);
    printf("%lld\n",CRT());
}
```

Part 3. 同余相关

扩展中国剩余定理

概念

当如下的一元线性同余方程组的模数不互质时，应使用扩展中国剩余定理进行求解 (Extended Chinese Remainder Theorem)。

$$\begin{cases} x \equiv a_1 (mod\ m_1) \\ x \equiv a_2 (mod\ m_2) \\ \dots \\ x \equiv a_k (mod\ m_k) \end{cases}$$ 。

Part 3. 同余相关

扩展中国剩余定理

求解过程

先看两个方程的情况：$x \equiv a_1 (mod\ m_1), x \equiv a_2 (mod\ m_2)$。
将其化为不定方程：$x = m_1 p + a_1 = m_2 q + a_2$，其中 $p, q \in \mathbb{Z}$。
由上式可得 $m_1 p - m_2 q = a_2 - a_1$，当 $gcd(m_1, m_2) \nmid a_2 - a_1$ 时无解，$gcd(m_1, m_2) \mid a_2 - a_1$ 时可用扩展欧几里得算法求解 p, q。
所以上述两个方程组成的同余方程组的解为 $x \equiv m_1 p + a_1 (mod\ M)$，其中 $M = lcm(m_1, m_2)$。
因此我们通过 $exgcd$ 算法将两个方程转化为了一个方程，当遇到含有 k 个一元同余线性方程组时，只需两两进行合并，做 $k - 1$ 次 $exgcd$ 算法即可。

Part 3. 同余相关

例题

luogu P4777 【模板】扩展中国剩余定理 (EXCRT)

给定 n 组非负整数 a_i, b_i，求解如下的关于 x 的同余方程组的最小非负整数解。

$$\begin{cases} x \equiv b_1 (mod\ a_1) \\ x \equiv b_2 (mod\ a_2) \\ \ldots \\ x \equiv b_k (mod\ a_k) \end{cases},$$

其中 $1 \le n \le 10^5, 1 \le a_i, b_i \le 10^{12}, lcm(a_1, a_2, ..., a_n) \le 10^{18}$。

Part 3. 同余相关

例题

Solution

```cpp
#include<bits/stdc++.h>
using namespace std;
#define N int(1e5+10)
#define reg register
typedef long long ll;
inline void read(ll &x){
    ll s=0,w=1;char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
    while(ch>='0'&&ch<='9'){s=(s<<3)+(s<<1)+(ch&15);ch=getchar();}
    x=s*w;
}
inline ll slowmul(ll x, ll y, ll m){
    while(x<0)x+=m;
    while(y<0)y+=m;
    if(x<y)swap(x,y);
    ll ret=0;
    while(y){
        if(y&1)(ret+=x)%=m;
        (x<<=1)%=m,y>>=1;
    }
    return ret;
}
```

```cpp
void exgcd(ll a, ll b, ll &x, ll &y){
    if(!b){x=1,y=0;return;}
    exgcd(b,a%b,y,x);y-=(a/b)*x;
}
ll n,ai[N],bi[N];
ll exCRT(){
    ll x,y,a,b,c,M=bi[1],ans=ai[1],gcd;
    for(ll i=2;i<=n;i++){
        a=M,b=bi[i],c=(ai[i]-ans%b+b)%b,gcd=__gcd(a,b);
        if(c%gcd)return -1;
        a/=gcd,b/=gcd,c/=gcd;
        exgcd(a,b,x,y);
        x=slowmul(x,c,b),ans+=x*M,M*=b;
        ans=(ans%M+M)%M;
    }
    return ans;
}
int main(){
    read(n);
    for(int i=1;i<=n;i++)read(bi[i]),read(ai[i]);
    printf("%lld\n",exCRT());
}
```

数学基础

Part 4. 组合数学

加法原理与乘法原理

加法原理

完成一个工作有 n 类方法，第 i 类方法的数目为 a_i, 那么完成这个工作有 $S = a_1 + a_2 + \dots + a_n$ 种不同的方法。

乘法原理

完成一个工作有 n 个步骤，第 i 个步骤的方法数有 a_i 个，那么完成这个工作有 $S = a_1 \times a_2 \times \dots \times a_n$ 种不同的方法。

Part 4. 组合数学

简单计数问题

不重复排列数

从 n 个不同元素的集合中，取 m 个元素排成一列，每取完一个元素不将其放回集合中，求排列的个数。

可重复排列数

从 n 个不同元素的集合中，取 m 个元素排成一列，每取完一个元素再将其放回集合中，求排列的个数。

不重复组合数

从 n 个不同元素的集合中，取 m 个元素组成一个新集合，每取完一个元素不将其放回原集合中，求新集合的个数。

可重复组合数

从 n 个不同元素的集合中，取 m 个元素组成一个新集合，每取完一个元素再将其放回原集合中，求新集合的个数。

Part 4. 组合数学

排列与组合

排列数

从 n 个不同元素中，任取 m 个元素按照一定的顺序排成一列，叫做从 n 个不同元素中取出 m 个元素的一个排列，这些排列的个数记为 A_n^m 或 P_n^m。

组合数

从 n 个不同元素中，任取 m 个元素组成一个集合，叫做从 n 个不同元素中取出 m 个元素的一个组合，这些组合的个数记为 $\binom{n}{m}$ 或 C_n^m 或 $C(m, n)$。

Part 4. 组合数学

性质

排列数

- $A_n^m = \frac{n!}{m!}$
- $nA_{n-1}^{m-1} = mA_n^m$
- $mA_{n-1}^{m-1} + A_{n-1}^m = A_n^m$

组合数

- $C_n^m = \frac{n!}{m!(n-m)!}$
- $C_n^m = C_n^{n-m}$
- $C_{n+1}^m = C_n^m + C_n^{m-1}$

Part 4. 组合数学

代码实现

引题

给定 n, m，求 $C_n^m \ mod \ 10^9 + 7$, 其中 $0 \le n, m \le 10^4$

实现

预处理

```
int main(){
    frac[0]=1LL,inv[0]=1LL;
    for(int i=1;i<=1000;i++)frac[i]=frac[i-1]*i%mod;
    inv[1000]=qpow(frac[1000],mod-2);
    for(int i=999;i;i--)inv[i]=inv[i+1]*(i+1)%mod;
}
```

短小精悍的代码

```
ll c(ll m, ll n){
    return frac[n]*inv[m]%mod*inv[n-m]%mod;
}
```

数学基础

Part 4. 组合数学

二项式定理

内容

$(a+b)^n = \sum_{i=0}^{n} C_n^i a^{n-i} b^i = C_n^0 a^n + C_n^1 a^{n-1} b + \cdots + C_n^n b^n$。

推论

- $\sum_{i=0}^{n} C_n^i = 2^n$
- $\sum_{i=0}^{n} (-1)^i C_n^i = 0$
- $\sum_{i=0}^{n} 2^i C_n^i = 3^n$

Part 4. 组合数学

抽屉原理

概念

将 $n+1$ 个物体划分为 n 组，那么至少有一组有两个 (或以上) 的物体。

推论

将 n 个物体划分为 k 组，那么至少存在一个分组含有 $\lceil \frac{n}{k} \rceil$(或以上) 的物体。

应用

抽屉原理 (鸽巢原理) 被常用于证明存在性证明和求最坏情况下的解。

Part 4. 组合数学

容斥原理

引题

班里有 36 个同学，20 个喜欢 dp,20 个喜欢图论，问有几个同学既喜欢 dp 又喜欢图论。答案显然是 4 个。
反过来想，设班里喜欢 dp 的同学集合为 A，喜欢图论的同学集合为 B，那么至少喜欢二者中一个的同学集合为 $A \cup B$。
则二者之间存在这样的数量关系 $|A \cup B| = |A| + |B| - |A \cap B|$。

推广

先推广到三个集合的情况，班里喜欢用洛谷的同学集合为 A，喜欢用 codeforces 的同学集合为 B，喜欢用 TJUOJ 的同学集合为 C，假设班里的同学至少喜欢一个网站，即班里所有同学集合为 $A \cup B \cup C$。
则存在这样的数量关系:
$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|$。

Part 4. 组合数学

容斥原理

定义

$|\bigcup_{i=1}^{n} A_i| = \sum_{i}^{n} |A_i| - \sum_{i,j:1<=i<j<=n} |A_i \bigcap A_j| + \sum_{i,j:1<=i<j<k<=n} |A_i \bigcap A_j \bigcap A_k| \ldots + (-1)^{n-1} \times |A_1 \bigcap \ldots \bigcap A_n|$

Part 4. 组合数学

例题

AtCoder Beginner Contest 297F-Minimum Bounding Box 2

第二场个人赛的 F 题，题目大意是在一个 $n \times m$ 的方格中随便挑选 k 个小矩形，用一个大矩形刚好覆盖这些小矩形，所得分数为这个大矩形的大小，问所有情况得分的总期望为多少。

Part 4. 组合数学
例题

Solution

我们不妨把问题换过来想，对于一个大矩形，它能对答案有多少贡献，答案是 $f_{i,j} * i * j * p_{i,j}$，其中 $f_{i,j}$ 代表矩形中包围 k 个小矩形的方案数,$p_{i,j}$ 代表该大矩形出现的概率。
相对来说 $p(i,j)$ 更好算，所有情况为 C_{mn}^{k}，矩形出现的次数为 $(n-i+1)(m-j+1)$，因此 $p(i,j)=\frac{(n-i+1)(m-j+1)}{C_{mn}^{k}}$。

Part 4. 组合数学

例题

Solution

对于 $f(i,j)$，我们利用容斥原理，先在大矩形里随便取 k 个，并排除不可行的方案，即未完全包围的方案。
大矩形中第一列、第一行、最后一列、最后一行中有空的，那么代表该情况不可行，
设 a_1, a_2, a_3, a_4 为缺第一行、第一列、最后一行、最后一列的情况，a_{12} 为缺第一行和第一列，以此类推
则 $f(i,j) = C_{ij}^k - a_1 - a_2 - a_3 - a_4 + a_{12} + a_{13} + a_{14} + a_{23} + a_{24} + a_{34} - a_{123} - a_{234} - a_{341} - a_{412} + a_{1234}$
$= C_{ij}^k - 2C_{(i-1)j}^k - 2C_{i(j-1)}^k + C_{(i-2)j}^k + C_{i(j-2)}^k + 4C_{(i-1)(j-1)}^k - 2C_{(i-1)(j-2)}^k - 2C_{(i-2)(j-1)}^k + C_{(i-1)(j-1)}^k$。
最后答案为
$ans = \sum_{i=1}^n \sum_{j=1}^m f(i,j)p(i,j)ij = \frac{\sum_{i=1}^n \sum_{j=1}^m ij(n-i+1)(m-j+1)f(i,j)}{C_{nm}^k}$。

数学基础

Part 4. 组合数学

卢卡斯定理

引题

给定 n, m, p，其中 p 为质数，求 $C_n^m \bmod p$， $n, m \le 10^9, p \le 10^5$。
n, m 是 10^9 范围的，这样再进行朴素求解将会消耗大量时间。

卢卡斯定理

卢卡斯定理: 对于质数 p, 有 $C_n^m \bmod p = C_{\lfloor \frac{n}{p} \rfloor}^{\lfloor \frac{m}{p} \rfloor} \times C_{n \bmod p}^{m \bmod p} \bmod p$。

Part 4. 组合数学

代码实现

P3807 【模板】卢卡斯定理/Lucas 定理

给定 n, m, p，其中 p 为质数，求 $C_n^m \bmod p$，$n, m, p \leq 10^5$。

代码实现

```
ll c(ll m, ll n,ll p){
    if(m>n||n==0)return 0;
    if(m==0)return 1;
    return frac[n]*inv[m]%p*inv[n-m];
}
ll lucas(ll n,ll m,ll p)
{
    if(n<p&&m<p)return c(n,m,p);
    return c(n%p,m%p,p)*lucas(n/p,m/p,p)%p;
}
```

Part 4. 组合数学

扩展卢卡斯定理

引题

给定 n, m, p，求 $C_n^m \bmod p$， $n, m \leq 10^{18}, p \leq 10^6$。不保证 p 为质数

扩展卢卡斯定理

当模数不为质数时，就要用到扩展卢卡斯定理，扩展卢卡斯定理和卢卡斯定理基本没什么关系，它也不是个定理，是个算法，前置知识主要为扩展欧几里得和中国剩余定理。

Part 4. 组合数学

过程一

中国剩余定理

我们要计算 $C_n^m \ mod \ M$，M 可能为合数。
根据唯一分解定理，$M = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_r^{\alpha_r}$，其中 $(p_i^{\alpha_i}, p_j^{\alpha_j}) = 1$。

因此我们可以构造 r 个同余方程：$\begin{cases} a_1 \equiv C_n^m (mod \ p_1^{\alpha_1}) \\ a_2 \equiv C_n^m (mod \ p_2^{\alpha_2}) \\ \dots \\ a_r \equiv C_n^m (mod \ p_r^{\alpha_r}) \end{cases}$，

计算出 $a_1, a_2, \dots, a_r$ 后，考虑中国剩余定理的过程，即可求出 $C_n^m \ mod \ M$。

Part 4. 组合数学

过程一

实现

```
ll exlucas(ll m, ll n, ll P){
  int cnt=0;
  ll p[20],a[20];
  for (ll i=2;i*i<=P;i++) {
    if (P%i==0){
      p[++cnt]=1;
      while(P%i==0)p[cnt]=p[cnt]*i,P/=i;
      a[cnt]=multilucas(m,n,i,p[cnt]);
    }
  }
  if(P>1)p[++cnt]=P,a[cnt]=multilucas(m,n,P,P);
  return CRT(cnt,a,p);
}
```

Part 4. 组合数学

过程二

计算 $C_n^m \ mod \ p_i^{\alpha_i}$

现在的问题转化为求 $C_n^m \ mod \ p_i^{\alpha_i}$。$C_n^m \ mod \ p_i^{\alpha_i} = \frac{n!}{m!(n-m)!} \ mod \ p_i^{\alpha_i}$。由于 $m!$ 与 $(n-m)!$ 不保证和 p_i 互质，所以无法直接求出 $m!$ 和 $(n-m)!$ 的逆元来计算上式。

因此将上式转化为 $\frac{\frac{n!}{p_i^x}}{\frac{m!}{p_i^y}\frac{(n-m)!}{p_i^z}} p_i^{x-y-z} \ mod \ p_i^{\alpha_i}$

将 $n!$ 做如下拆分：

$n! = 1 \times 2 \times \ldots \times n = (p \times 2p \times \ldots \times tp) \times (1 \times 2 \times \ldots \times p-1 \times p+1 \times \ldots \times n)$

$= p^t \times (1 \times 2 \times \ldots \times t) \times (\prod_{i,(i,p)=1}^{n} i) = p^t \times t! \times (\prod_{i,(i,p)=1}^{p^k} i)^{\lfloor \frac{n}{p^k} \rfloor} \times (\prod_{i,(i,p)=1}^{n \ mod \ p^k} i)$

其中 $t = \lfloor \frac{n}{p} \rfloor$，观察到等式两侧只有 $n!, p^t, t!$ 含有因子 p，则

$\frac{n!}{p^x} = (\frac{t!}{p^y})(\prod_{i,(i,p)=1}^{p^k} i)^{\lfloor \frac{n}{p^k} \rfloor}(\prod_{i,(i,p)=1}^{n \ mod \ p^k} i)$

递归求解第一项，循环暴力求解后两项即可。

Part 4. 组合数学

过程二

代码实现

```cpp
ll calc(ll n, ll x, ll P) {
  if(!n)return 1;
  ll s=1;
  for (ll i=1;i<=P;i++)
    if(i%x)s=s*i%P;
  s=qpow(s,n/P,P);
  for (ll i=n/P*P+1;i<=n;i++)
    if(i%x)s=i%P*s%P;
  return s*calc(n/x,x,P)%P;
}
ll multilucas(ll m, ll n, ll x, ll P) {
  int cnt=0;
  for (ll i=m;i;i/=x)cnt+=i/x;
  for (ll i=n;i;i/=x)cnt-=i/x;
  for (ll i=m-n;i;i/=x)cnt-=i/x;
  ll x1=qpow(x,cnt,P)%P*calc(m,x,P)%P;
  ll x2=inv(calc(n,x,P),P)%P*inv(calc(m-n,x,P),P)%P;
  return x1*x2%P;
}
```

基础数学

Thanks for Listening!

QQ:407694747

blog:https://blog.csdn.net/dhdhdhx