数论

吴清月

数论

- 整除
  - 整除与带余除法
  - 欧几里得算法
  - 质数与质因数分解
- 同余理论
  - 同余关系
  - 费马小定理
  - 欧拉定理与扩展欧拉定理
  - 线性同余方程组
  - 二次剩余
  - 离散对数
- 数论函数
  - 数论函数
  - 常见数论函数
  - 欧拉筛
  - 数论分块
  - 狄利克雷卷积
  - 莫比乌斯反演
  - 杜教筛
- 扩展埃氏筛
  - 洲阁筛
  - min25筛

Part I

整除

整除

- **定义**
  - 对于两个正整数p和q，若存在正整数x，满足$px = q$，则称p整除q，记作$p|q$。
- **整除的相关性质：**
  - 自反性：$p|p$
  - 传递性：$p|q, q|r \to p|r$
  - 反对称性：$p|q, q|p \to p = q$

带余除法

- 定义
  - 对于正整数a, b，存在唯一一对整数(q, r)，满足$a = bq + r$且$0 \leq r < b$。称q为商，r为余数。
- 取整
  - 下取整：$\lfloor x \rfloor = \max\{k \in Z | k \leq x\}$
  - 上取整：$\lceil x \rceil = \min\{k \in Z | k \geq x\}$
- 可以发现$q = \lfloor \frac{a}{b} \rfloor$
- 取模
  - 对于正整数a, b，定义模运算为
  - $a \bmod b = a - \lfloor \frac{a}{b} \rfloor b$
- 注意在C++中，除法返回的结果是向0取整。

欧几里得算法

- 最大公约数与最小公倍数

$$\gcd(x, y) = \max\{v | (v|x) \land (v|y)\}$$
$$\operatorname{lcm}(x, y) = \min\{v | (x|v) \land (y|v)\}$$

- 欧几里得算法：
  - 当$y \neq 0$时，有$\gcd(x, y) = \gcd(y, x \bmod y)$

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

扩展欧几里得算法

- **裴蜀定理**
  - 不定方程$ax+by=c$有解，当且仅当$\gcd(x,y)|c$。
- **证明&求解：**
  - 必要性显然
  - 充分性证明可以考虑在求解gcd的时候递归构造解。

$$ax+by=c$$
$$b'y+a'(x \bmod y)=c$$
$$a'\left(x-\left\lfloor\frac{x}{y}\right\rfloor y\right)+b'y=c$$
$$a'x+\left(b'-a'\left\lfloor\frac{x}{y}\right\rfloor\right)y=c$$

  - 每一次令$a=a',b=b'-a'\left\lfloor\frac{x}{y}\right\rfloor$。边界条件是当$y=0$时，$a=\frac{c}{x},b=0$。

扩展欧几里得算法——代码

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

质数

- **定义**
  - 对于一个大于1的整数x，如果它不被任何$1 < y < x$的整数y整除，那么称x为质数或素数，否则称x为合数。
- **定理**
  - 质数有无穷多个。
  - 证明：若有有限个质数$p_1 \dots p_m$，则令$n = \prod_{i=1}^{m} p_i$，则$n + 1$不能被$p_1 \dots p_m$整除，所以$n + 1$也是质数。
- **设$\leq n$的质数有$\pi(n)$个，则$\pi(n) = O\left(\frac{n}{\log n}\right)$。**
- **算数基本定理**
  - 任何正整数都能唯一表示成一些质数的幂的乘积。

埃氏筛

- 对于每一个质数，筛掉其倍数。
- 时间复杂度$O(n\log\log n)$。
- 定理：

$$\sum_{i=1}^{n}\frac{1}{i}=O(\log n)$$

$$\sum_{i=1}^{n}\frac{1}{p_i}=O(\log\log n)$$

（不得不说ppt插入公式真的丑）

埃氏筛

代码

```
for(int i=2;i<=n;i++)
{
    if(!flag[i])prime[++num]=i;
    for(int j=1;j<=num;j++)
    {
        if(i*prime[j]>n)break;
        flag[prime[j]*i]=1;
    }
}
```

欧拉筛

- 埃氏筛还是不够优秀！
- 我们想要得到一个$O(n)$的算法。
- 在埃氏筛中，我们发现了这样一个问题：每一个合数会被筛很多次，比如30这个数就被2,3,5各筛了一次。
- 能不能优化这个过程呢？换句话说，能不能使得每一个合数都只被筛一次呢？
- 答案是肯定的。

欧拉筛

- 我们尝试让一个合数只在枚举到它最小的质因子的时候被筛掉。
- 回顾埃氏筛的过程，当我们到达i时，我们顺次枚举当前的所有质数，并且筛掉这个质数乘i。
- 因为我们要让这个质数是筛掉的合数的最小质因子，所以如果i这个数包含p这个质因子，那么大于p的所有质数就不用枚举了，因为如果继续枚举得到的数最小质因子都会小于当前枚举的质数。
- 比如现在我们有2,3,5这三个质数，现在i是6，那么我们枚举到2的时候发现2是6的质因子，所以后面的3和5都不用枚举了，因为 $3 \times 6, 5 \times 6$ 的最小质因子是2，而不是3和5。

欧拉筛——代码

```
flag[1]=1;
for(int i=2;i<=n;i++)
{
      if(!flag[i])prime[++num]=i;
      for(int j=1;j<=num&&prime[j]*i<=n;j++)
      {
            flag[i*prime[j]]=1;
            if(i%prime[j]==0)break;
      }
}
```

素性测试

- 定义法：$O(n)$。
- 试除法：$O(\sqrt{n})$ 或 $O\left(\frac{\sqrt{n}}{\log n}\right)$
- Miller_Rabin：$O(k\log n)$

Miller_Rabin素性测试

费马小定理

- 费马小定理：$a^{p-1} \equiv 1 \pmod p$
- 既然费马小定理成立，那么费马小定理的逆定理成立吗？

$$\forall a \in [1, p-1], a^{p-1} \equiv 1 \pmod p \rightarrow p\text{是质数？}$$

- 不一定。
- 对于任何和n互质的数，$a^{n-1} \equiv 1 \bmod n$的合数被称为Carmichael Number
- 最小的Carmichael Number是561=11*51，显然这个算法错误率很高。

Miller_Rabin素性测试

二次探测定理

- 另一个定理
  - 若p为质数，则对于任意的正整数x，若
$$x^2 \equiv 1 \pmod p$$
  - 则
$$x \equiv 1$$
  - 或
$$x \equiv p-1$$
- 这个定理的逆定理是成立的，可以用这个来判定素数
- 找若干个满足$x^2 \equiv 1$的整数x，检查x是否为1或$p-1$

Miller_Rabin素性测试

$$a^{p-1} \equiv 1 \pmod p$$

- a^{p-1}就是一个满足条件的x^2!
- 我们可以将指数不断除以2，直到a^k变成$p-1$。当然也有可能一直是1。
- 令$p-1 = k \cdot 2^r, t_i = k \cdot 2^i$，则若$a^{t_i} \equiv 1$，则$a^{t_{i-1}} \equiv 1$或$a^{t_{i-1}} \equiv p-1$
- 我们可以先用快速幂求出t_0，然后每次求平方，如果它变成1了而上一个不是$p-1$，或者乘到最后它也没变成1，则p是一个合数
- 根据某玄学定理，单词探测正确率为$\frac{3}{4}$，实际应用中只需要探测前几个质数。

Miller_Rabin素性测试——代码

```
bool check(int p,ll n)
{
    if(p==n)return 1;
    if(quick_pow(p,n-1,n)!=1)return 0;
    ll s=0,k=n-1;
    while(!(k&1))k>>=1,s++;
    ll last=quick_pow(p,k,n);
    for(int i=1;i<=s;i++)
    {
        ll now=last*last%n;
        if(now==1)return last==n-1||last==1;
        last=now;
    }
    return 0;
}
```

Pollard_Rho质因数分解

- 讲了Miller_Rabin怎么能不讲Pollard_Rho呢
- 对于一个合数n，尝试找到一个因子d，递归分解d和$\frac{n}{d}$，若n通过了素性测试那就停止分解。
- 那么怎么找到这个因子呢?
  - 直接随机？期望$O(n)$
  - 随机一个数然后与n求gcd？$O(\sqrt{n})$（为啥我们不直接暴力呢）

Pollard_Rho质因数分解

- 设p是n最小的质因子，如果我们找到了两个数a, b，满足$a \equiv b \pmod p$但是$a \neq b$，那么$\gcd(n, |a-b|)$一定是p的倍数。
- 根据生日悖论，每次随机一个数x，期望随机$O\left(\sqrt{p}\right)$次，就能找到两个数字模p同余。
- 但是这样我们需要存储$O\left(\sqrt{p}\right) = O\left(n^{\frac{1}{4}}\right)$个数~~（这公式有毒）~~，内存开不下。
- 于是Floyd告诉我们，只需要存两个数就够了！

Pollard_Rho质因数分解

Floyd判圈法

- 一般情况下，我们的伪随机数生成器是$x = x_0^2 + a \bmod n$，这样的生成器生成的伪随机数一定是成环的。
- 想象两个小孩在一个环上跑步，第一个小孩一次跑一步，第二个小孩一次跑两步，那么这两个小孩一定会相遇。
- 所以，我们可以每次令$x_1 = f\big(f(x_1)\big), x_2 = f(x_2)$，这样当两个数字的差与$n$的$\gcd \neq 1$，即这两个数模$p$相等时，我们便找到了$n$的一个约数。
- 由生日悖论得出，找到一个约数的时间复杂度是$O\left(n^{\frac{1}{4}}\right)$的。
- 特别需要注意的是这两个数字模n同余的时候，这时候需要重新随机。

Pollard_Rho质因数分解——代码

```
void Pollard_Rho(ll x)
{
    if(Miller_Rabin(x))return;
    while(1)
    {
        ll a=rand()%x,v1=a,v2=f(a,x);
        while(v1!=v2)
        {
            ll d=__gcd(x,(v2-v1+x)%x);
            if(d>1&&d!=x)
            {
                Pollard_Rho(max(d,x/d));
                Pollard_Rho(min(d,x/d));
                return;
            }
            v1=f(v1,x),v2=f(f(v2,x),x);
        }
    }
}
```

Part II

同余理论

预知识——群

- 定义
- 满足以下四个性质的集合$(S,\cdot)$叫做群：
  - 封闭性：定义一种乘法运算$\cdot$，使得$\forall a,b \in S$，$a \cdot b \in S$。
  - 结合律：$(a \cdot b) \cdot c = a \cdot (b \cdot c)$
  - 单位元：$\exists e \in S, \forall a \in S, a \cdot e = a$
  - 逆元：$\forall a \in S, \exists a^{-1} \in S, a \cdot a^{-1} = e$
- 举例：$\{-1,1\}, Q^*, R^*$
- 群的性质：
  - 单位元唯一，若存在两个单位元e_1, e_2，则$e_1 = e_1 e_2 = e_2$。
  - 逆元唯一，若a存在两个逆元b, c，则$b = b \cdot (a \cdot c) = (b \cdot a) \cdot c = c$

同余关系

- 一个不能被p整除的整数n，对一个质数p取模后，有多少种情况呢?

$$1,2,3,\ldots,p-1$$

- 这些数字叫做模p意义下的剩余系。
- 然后我们惊奇的发现，这$p-1$个数字构成了一个群!
  - 乘法运算定义为模意义下的乘法
  - 满足结合律
  - 单位元是1
  - 对于任意一个数，都有乘法逆元

费马小定理

- 既然任意一个数都存在逆元，那么我们很自然的想到，如何求出一个数的逆元
  - 方法一：exgcd，等价于求$ax - bp = 1$的解
  - 方法二：费马小定理。证明用上面讲的群

$$\forall a \in [1, p-1], a^{p-1} \equiv 1 \pmod{p}$$

$$a^{p-2} \cdot a \equiv 1 \pmod{p}$$

- 也就是说，a^{p-2}就是a的乘法逆元。
- 快速幂大家都会写吧？

预知识——快速幂

```
ll quick_pow(ll x,ll a,ll MOD)
{
    ll ans=1;
    while(a)
    {
        if(a&1)ans=ans*x%MOD;
        x=x*x%MOD;
        a>>=1;
    }
    return ans;
}
```

欧拉定理

- 对于一个整数n，我们设欧拉函数$\varphi(n)$表示1到n中和n互质的数的个数，比如$\varphi(3)=2,\varphi(7)=6$。
- 对于一个模数p，$a^{\varphi(p)} \equiv a^{2\varphi(p)} \bmod p$。如果$(a,p)=1$，那么$a^{\varphi(p)} \equiv 1 \bmod p$。
- 扩展一下：
- $a^b \equiv \begin{cases} a^{b \bmod \varphi(p)} & \text{if } (a,p)=1 \\ a^{b \bmod \varphi(p)+\varphi(p)} & \text{if } b \ge \varphi(p) \\ a^b & \text{if } b < \varphi(p) \end{cases}$

线性同余方程组

- 我们要求解这样一组同余方程的解：
- $$\begin{cases} x \equiv a_1 \bmod m_1 \\ x \equiv a_2 \bmod m_2 \\ \qquad\vdots \\ x \equiv a_k \bmod m_k \end{cases}$$
- 我们的目标是解出来$x \equiv a \bmod m$。
- 考虑每次合并两个同余方程得到一个新的同余方程，这样合并$k-1$次后就可以得到整个方程组的解。
- 接下来我们讨论如何合并两个同余方程。

线性同余方程组

- $\begin{cases} x \equiv a_1 \bmod m_1 \\ x \equiv a_2 \bmod m_2 \end{cases}$
- $\begin{cases} x - k_1 m_1 = a_1 \\ x - k_2 m_2 = a_2 \end{cases}$
- $a_1 + k_1 m_1 = a_2 + k_2 m_2$
- $k_1 m_1 - k_2 m_2 = a_2 - a_1$
  - $ax + by = c$
- 用扩展欧几里得算法即可求解。
- $a = a_1 + k_1 m_1, m = \operatorname{lcm}(m_1, m_2)$
- 将这个过程重复$k-1$次即可。

二次剩余

- 如果一个数x可以在模p意义下开根，那么我们称x是一个模p意义下的二次剩余。
- 判断方式：若$x^{\frac{p-1}{2}} \equiv 1 \bmod p$，那么$x$是模$p$意义下的二次剩余。
- 对于一个奇质数p，假设a是一个二次剩余，那么一定存在两个u, v，满足$u^2 \equiv a, v^2 \equiv a$。

二次剩余

Cipolla 算法

- 如何求x满足$x^2 \equiv n \pmod p$?
- 下面介绍一个我也不知道怎么想出来的算法：
- 首先随机一个数a满足$(a^2-n)^{\frac{p-1}{2}} \equiv -1 \pmod p$
- 期望次数？2次！
- 设$\omega = \sqrt{a^2-n}$，由于不存在，扩域设其为虚根。
- 所谓扩域，就是把所有的数都写成$a\omega + b$的形式，此时一个数不再是int，而是pair<int,int>。
- 则$(a+\omega)^{\frac{p+1}{2}}$是一个符合条件的$x$。

二次剩余

Cipolla 算法

- 证明：
- $x^2 \equiv (a+\omega)^{p+1}$
- $\equiv (a+\omega)^p(a+\omega)$
- $\equiv (a+\omega)\left(\sum_{i=0}^{p} a^i \omega^{p-i}\binom{p}{i}\right)$
- 注意到后面有一个组合数，而除了$\binom{p}{0}$和$\binom{p}{p}$以外别的组合数分子上都有一个p，所以都是0。
- 所以原式继续化简为$(a+\omega)(a^p+\omega^p)$
- $\omega^{p-1} \equiv -1$
- 所以原式$\equiv (a+\omega)(a-\omega) \equiv a^2-\omega^2 \equiv n$
- 这种算法记住结论就好啦

离散对数

BSGS

- 求解一个x，满足$a^x \equiv b \pmod p$，且a, b与p互质。
- 我们考虑用分块的思想。
- 先设一个块大小S，我们预处理出$a^0, a^1, a^2, \ldots, a^{S-1}$
- 然后我们再用快速幂处理出$a^S, a^{2S}, a^{3S}, \ldots, a^{\frac{p}{S}\cdot S}$
- 把第一次预处理出来的所有数丢到一个hash表里面，然后枚举第二次预处理出来的数。
- 假设我们枚举到了a^{kS}，那么我们检查$\frac{b}{a^{kS}}$是否在hash表里，如果在我们就找到了一组解。
- 时间复杂度$O\left(S + \frac{p}{S}\right)$，令$S = \sqrt{p}$可得复杂度$O(\sqrt{p})$。

离散对数

exBSGS

- 如果a,b不和p互质，那么问题在于无法求出来逆元。
- 想办法让它们互质。
- 令$d=\gcd(a,p)$，则如果d不整除b方程无解，否则两边同时除以d：
- $\frac{a}{d}\cdot a^{x-1}\equiv\frac{b}{d}\left(\bmod \frac{p}{d}\right)$
- 不断递归这个过程直到a与p互质。
- 则方程最终变成了$a_0 a^{x-k}\equiv b_0 (\bmod p_0)$
- 此时a_0和p_0互质，除过去直接BSGS即可。
- 注意需要特判$x<k$的情况。这部分可以暴力枚举x。

Part III

数论函数

数论函数

- **数论函数可以理解为定义域是正整数的函数。**
- **积性函数：**
  - 如果对于一个数论函数$f(x)$，满足对于任意$\gcd(a,b)=1$，都有$f(ab)=f(a)f(b)$，那么$f(x)$称作积性函数。
- **完全积性函数：**
  - 如果对于一个数论函数$f(x)$，满足对于任意a,b，都有$f(ab)=f(a)f(b)$，那么$f(x)$称作完全积性函数。

积性函数

- 如果$f(x)$和$g(x)$都是积性函数，那么下面这些函数也是积性函数：
  - $h(n) = f(n)g(n)$
  - $h(n) = f(n^p)$
  - $h(n) = \sum_{d|n} f(d)g\left(\frac{n}{d}\right)$
- 对于一个积性函数$f(n)$，我们只需要知道它在所有质数的幂的位置上的取值$f(p^e)$，就能知道所有的$f(n)$。
- 某些积性函数可以用线性筛求出来。

积性函数

- 一些常见的积性函数：
  - $\epsilon(n) = [n = 1]$
  - $1(n) = 1$
  - $id_k(n) = n^k$
  - $I(n) = id_1(n) = n$
  - $\sigma_k(n) = \sum_{d|n} d^k$
  - $\varphi(n) = \sum_{i=1}^{n} [\gcd(i, j) = 1]$
  - $\mu(n) = \begin{cases} 0 & (n\text{有平方质因子}) \\ (-1)^k & (n\text{可以写成}k\text{个质数的积}) \end{cases}$

欧拉筛

- 对于一个积性函数，如果$f(p^e)$有比较好的封闭形式，那么可以用欧拉筛来求，时间复杂度是$O(n)$的。
- 筛φ?
- 和p^e互质的数只要不包含p这个质因子就行了，所以$\varphi(p^e) = p^{e-1}(p-1)$，也就是$\varphi(p^e) = p\varphi(p^{e-1})$
- 筛μ?
- $\mu(p) = -1$，同时$\mu(p^e) = 0$（$e > 1$），也可以用欧拉筛。

欧拉筛

- 筛σ_0（约数个数）？
- 假设$n=\prod p_i^{e_i}$，那么$\sigma_0(n)=\prod(e_i+1)$。
- 对于每一个数，我们记录一下它的最小质因子$minp$和最小质因子的次数c。
- 在欧拉筛的过程中，假设我们枚举到了p和$\frac{n}{p}$，其中p是n的最小质因子，那么如果$\frac{n}{p}$的最小质因子也是p，则$c_n=c_{n/p}+1$，同时$\sigma_0(n)=\sigma_0\left(\frac{n}{p}\right)\cdot\frac{c_n+1}{c_{n/p}+1}$。
- 否则，$c_n=1,\sigma_0(n)=\sigma_0\left(\frac{n}{p}\right)\cdot 2$。

数论分块

- 什么是数论分块呢?
- 我们考虑我们需要求这么一个式子:
- $\sum_{i=1}^{n} f\left(\left\lfloor \frac{n}{i} \right\rfloor\right)$
- 其中$f(i)$可以$O(1)$求出来。
- 直接做是$O(n)$的。
- 如何优化这个过程呢?
- 我们假设$n = 12$,那么每一个$\left\lfloor \frac{n}{i} \right\rfloor$是下面这样的:
- i =1 2 3 4 5 6 7 8 9 10 11 12
- n/i=12 6 4 3 2 2 1 1 1 1 1 1
- 后面一堆2一堆1可不可以一起算呢?

数论分块

- 我们发现，从5到6得到的数字都是2，从7到12得到的数字都是1。
- 如果n/i是x，那么我们希望得到一个j，使得从n/i到n/j得到的数字都是x。
- 实际上，j=n/(n/i)。比如12/(12/7)=12/1=12。
- 所以我们for里面枚举i，每次让i=n/(n/i)+1。

数论分块

代码

```
for(int i=1;i<=n;)
{
    int j=n/(n/i);
    ans+=(j-i+1)*f(n/i);
    i=j+1;
}
```

数论分块

- 这样的时间复杂度是$O(\sqrt{n})$的。
- 为什么呢?
- 对于一个数i，要么$i \le \sqrt{n}$，要么$\left\lfloor \frac{n}{i} \right\rfloor \le \sqrt{n}$。
- 然而小于等于$\sqrt{n}$的数只会有根号个，所以这样做时间复杂度是$O(\sqrt{n})$的。

Q&A

- Thanks for listening!