同余性质在算法竞赛中的应用

同余（数学基础）

蒋世超

东北林业大学计算机科学与技术系

2021 年 8 月 3 日

- 我们不是数学家，对于公式定理会用就行

- 定义：
- 两个整数 a、b，若它们除以整数 m 所得的余数相等，则称 a 与 b 对于模 m 同余或 a 同余于 b 模 m
- 记作：$a \equiv b(mod\ m)$

- 欧拉定理：
- 若 a，n 互质，则：
- $a^{\varphi(n)} \equiv 1(mod\ n)$
- 亦可推得 $a^b \equiv a^{b\%\varphi(n)}(mod\ n)$
- 欧拉函数 $\varphi(n)$ 即：对正整数 n，小于 n 的正整数中与 n 互质的数的数目
- 欧拉函数是一个很重要的积性函数，若 i 与 p 互质，则：
- $\varphi(i * p) = \varphi(i) * \varphi(p)$
- 推论就是我们常说的欧拉降幂，他可以有效的降低指数的大小到我们可以计算的范围内，从而优化快速幂的计算

- 求欧拉函数：
- 通常是通过筛法去求 $\varphi(n)$，这一点并不复杂，只需要简单改动一下我们现有的素数筛即可
- 需要用到以下两个推论
- 若 $i \ mod \ p = 0$ 则 $\varphi(i * p) = \varphi(i) * p$
- 若 $i \ mod \ p \neq 0$ 则 $\varphi(i * p) = \varphi(i) * (p - 1)$

```
void getphi() {
    int i, j;
    phi[1] = 1;
    for (i = 2; i <= N; i++) {
        if (!vis[i]){
            prime[++tot] = i;
            phi[i] = i - 1;
        }
        for (j = 1; j <= tot; j++) {
            if (i * prime[j] > N)
                break;
            vis[i * prime[j]] = 1;
            if (i % prime[j] == 0) {
                phi[i * prime[j]] = phi[i] * prime[j]; //(1)
                break;
            }
            else
                phi[i * prime[j]] = phi[i] * (prime[j] - 1); //(2)
        }
    }
}
```

- 之后就还是套用快速幂的板子即可

- 费马小定理:
- 若 p 是质数,则对任意整数 a,都有:
- $a^p \equiv a(mod\ p)$
- 同样可以用来缩小快速幂的数据大小(注意与欧拉降幂的不同适用范围)
- 这个定理将会广泛的用在求逆元当中

- 逆元：
- 如果有一个线性同余方程 $ax \equiv 1(mod\ b)$，则称 x 为 $a\ mod\ b$ 的逆元，记作 a^{-1}
- 更加常见的表述如下：
- 求 $\frac{a}{b}$ 对 mod 取模，在这种情况下我们实际上是求 $a \times inv(b)\%mod$

- 求逆元之费马小定理:
- $ax \equiv 1(mod\ b)$
- $ax \equiv a^{b-1}(mod\ b)$
- $x \equiv a^{b-2}(mod\ b)$
- 由此便可以通过快速幂在 $O(log)$ 的复杂度内求逆元，但并不是所有情况下，这种情况都可以适用，尤其是对于需要求较多数的逆元的题目当中容易 TLE，此时我们需要使用线性法

- 求逆元之线性法：
- 证明较为繁琐，好懂但不易讲解，各位同学可以自行去搜索相关证明推导的过程，这里我们直接提供结论
- $i^{-1} = \begin{cases} 1 & when \;\; i = 1 \\ -\lfloor \frac{p}{i} \rfloor (p \; mod \; i)^{-1} & otherwise \end{cases} \pmod p$
- 需要注意的是，我们虽然可以线性的求出 1-n 的所有数的逆元，但事实上并非他们均具有逆元，使用时需要注意实际

```
inv[1] = 1;
for (int i = 2; i <= n; ++i) {
  inv[i] = (p - p / i) * inv[p % i] % p;
}
```

- 不过单有乘法逆元，可能并不能挑出一些好玩有趣的题目
- 组合数学是数论当中比较重要的一块，不妨就给同学们介绍一下卢卡斯定理的相关内容
- 相信同学们也已经发现了，在已有的知识体系内，组合数是个非常难处理的东西，因为如果直接计算会涉及到阶乘，而求阶乘的复杂度是 $O(n)$ 的，如果大量重复计算或者数据范围很夸张就很容易 T 掉
- 这种情况下就必须要请出 Lucas 定理

- 同样的，证明不在此讲解，我们直接介绍定理内容
- $C_n^m = lucas(n,m) = C_{n\ mod\ p}^{m\ mod\ p} \times lucas(\frac{n}{p}, \frac{m}{p})\ (mod\ p)$
- 实质是将大组合数取模拆分小组合数取模，而小组合数的计算用乘法逆元实现即可
- 需要注意的是，这个算法的时间复杂度一般是 $O(plog_pm)$ 的，不适用于模数很大的情况，此时应当使用传统的预处理阶乘及阶乘逆元直接套公式求解（但一般不会出现数又大，模数又大的情况，那也太不当人了）

```
//实质还是不断调用小费马定理求逆元去搞组合数
ll lucas(ll n, ll m, ll p) {
    if (n < m) {
        return 0;
    }
    if (m == 0) {
        return 1;
    }
    else {
        return (C(n % p, m % p, p) * lucas(n / p, m / p, p));
    }
}
```

- 逆元等均是求解问题的工具，他们本质不是算法，都是需要我们分析问题之后得到求解的目标之后再使用上述的各种数学工具
- 同余具有很多很好的性质，厉害的出题人经常组合这些性质并出题
- 组合数学的内容很多，这里只是简单介绍，题目选择的也不难，一共 7 题每天一题即可

1 说在最前

2 同余

3 参考文献

Thanks!