算法竞赛中的数学问题——讲义

东北育才学校　张昕海

February 5, 2018

Contents

1 快速幂

理论基础：

(1) 若 $x \equiv y \pmod p$，则 $x^n \equiv y^n \pmod p$；

(2) 二进制与十进制的相互转化、位运算.

代码： 求 $x^n \bmod p$.

```
int ans=1;
while(n){
    if(n&1) ans=ans*x%p;
    n>>=1;
    x=x*x%p;
}
```

思想： 我们把正整数 n 写成二进制的形式，如 $n=11=(1011)_2$，则我们可以把 x^{11} 拆成 $x^8 \cdot x^2 \cdot x^1$ 来计算.

具体实现： 我们用变量 a 保存中间结果 x^{2^i}，如果 n&1 就乘入 ans，否则不乘入 ans，同时指数 n 通过 `n>>=1` 来更新，中间结果 a 通过 `a=a*a` 来进行更新.

时间复杂度： $O(\log n)$.

特别提醒： 这是一个绝大多数 OI 选手都熟练掌握而且写得都差不多的算法. 如果在考试时使用，建议使用一些具有个人特色的变量名，防止被认定为**代码雷同**.

2 素数筛法

2.1 朴素的素数判断

代码：

```
for(int i=2; i<=n; i++){
    bool flag=1;
    for(int j=2; j<=sqrt(i+0.5); j++)
        if(i%j==0) flag=0;
    ...
}
```

时间复杂度： $O(n\sqrt{n})$，复杂得不得了.

原因： 对于某个正整数 i，它被小于等于 $\sqrt{i}$ 的每个正整数都试除了一遍——这是完全没有必要的.

改进： 对于每个正整数 i，尝试只用它的质因子去试除.

2.2 Eratosthenes 筛法

代码：

```
int m=sqrt(n+0.5);
memset(vst,0,sizeof(vst));
for(int i=2; i<=m; i++)
    if(!vst[i])    //!vst[i]表示i为素数.
        for(int j=i*i; j<=n; j+=i) vst[j]=1;
```

算法的思想：引入一个 vst 数组，每找到一个素数 i，就筛掉大于等于 i^2 的所有 i 的倍数（为什么不用考虑 i^2 之前 i 的倍数？），这样筛去合数的过程变得具有目的性.

时间复杂度：$O(n\log\log n)$，已经比较理想了.

继续优化的可能：事实上，对于每个正整数 i，它被它的所有质因子都筛掉了一遍——我们可以尝试让每个正整数 i 都只被它的某一个正整数筛掉.

2.3 线性筛法

代码：

```
bool check[MAXN]={0};
int prime[MAXN]={0};
int tot=0;
for(int i=2; i<=N; i++){
    if(!check[i]) prime[tot++]=i;
    for(int j=0; j<tot&&i*prime[j]<=N; j++){
        check[i*prime[j]]=1;
        if(i%prime[j]==0) break;    //关键在于理解此句.
    }
}
```

对于代码的解释：这里我们用 prime 数组来记录已经找到的所有素数，check 数组记录每个正整数是否为素数（记为 0）. 顺次扫描 $2\sim N$ 中的每个正整数，如果扫到某个正整数 i 的时候满足!check[i]，则 i 为素数，加入素数队列 prime. 同时，无论 i 是否为素数，都筛掉 $ip_0, ip_1, \cdots$，直到 $p_j \mid i$. 这样的话，我们就可以保证**每个正整数都只被它最小的素数筛掉**，从而一定程度地优化了算法.

时间复杂度：$O(n)$.

如何理解 $p_j \mid i$：如果 $p_j \mid i_0$，那么对于 p_j 后的任意 p_k，$p_j \mid i_0 p_k$，我们不妨让 $i_0 p_k$ 在 $i = \dfrac{i_0 p_k}{p_j} (> i_0)$ 时被 ip_j 筛掉. 这样就避免了重复筛掉同一个正整数. 事实上，这样做之后，每个正整数 i 都会被它的最小质因子筛掉.

拓展——求欧拉函数 $\varphi(n)$ 的值：

(1) 欧拉函数 $\varphi(n)$：对于任意正整数 n，我们用 $\varphi(n)$ 表示 $1\sim n$ 中与 n 互质的正整数的个数. 欧拉函数是一个非常重要的数论函数.

(2) 性质：

- 对于素数 p，$\varphi(p) = p-1$；
- 对于互质的正整数 x, y，$\varphi(xy) = \varphi(x)\varphi(y)$；
  特别地，若 $x = p$ 为质数，且 $p \nmid y$，则 $\varphi(py) = \varphi(p)\varphi(y) = (p-1)y$；
- 对于素数 p 和与它不互质的正整数 x，$\varphi(px) = p\varphi(x)$；
- 对于任意正整数 $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_n^{\alpha_n}$，$\varphi(n) = n\left(1-\dfrac{1}{p_1}\right)\left(1-\dfrac{1}{p_2}\right)\cdots\left(1-\dfrac{1}{p_n}\right)$.

(3) 代码：只需对线性筛法的代码稍作改动.

```
bool check[MAXN]={0};
int prime[MAXN]={0};
int tot=0;
int phi[MAXN]={0};
phi[1]=1;    //不要忘记.
for(int i=2; i<=N; i++){
    if(!check[i]){
        prime[tot++]=i;
```

```
        phi[i]=i-1;
    }
    for(int j=0; j<tot&&i*prime[j]<=N; j++){
        check[i*prime[j]]=1;
        if(i%prime[j]==0){
            phi[i*prime[j]]=phi[i]*prime[j];
            break;
        }
        else phi[i*prime[j]]=phi[i]*(prime[j]-1);
    }
}
```

(4) 时间复杂度：同线性筛法，$O(n)$.

3 费马小定理与欧拉定理

3.1 费马小定理

叙述：已知正整数 a 和质数 p，且 $\gcd(a,p)=1$，则 $a^{p-1} \equiv 1 \pmod p$.

应用：结合快速幂求乘法逆元，即 a 模 p 的乘法逆元为 a^{p-2}.

3.2 欧拉定理

叙述：若正整数 a,n 互质，则 $a^{\varphi(n)} \equiv 1 \pmod n$.

广义欧拉定理：对于任何整数 x 和 $n \geqslant \varphi(k)$，则 $x^n \equiv x^{n\%\varphi(k)+\varphi(k)} \pmod k$.

4 欧几里得除法

4.1 普通的欧几里得除法

问题：已知正整数 a,b，求 a,b 的最大公约数.

理论基础：$\gcd(a,b)=\gcd(b,a\%b)$，我们可以通过有限次这样的操作求出 $\gcd(a,b)$.

代码：

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

4.2 扩展欧几里得除法

问题：已知正整数 a,b 和它们的最大公约数 d，求方程 $ax+by=d$ 的一组整数解 (x,y).

理论基础：(裴蜀定理) 若正整数 a,b 满足 $\gcd(a,b)=d$，则必存在整数 x,y 满足 $ax+by=d$.

特别地，正整数 a,b 互质的充要条件为方程 $ax+by=1$ 有整数解 (x,y).

代码：

```
void gcd(int a,int b,int& d,int& x,int& y){
    if(b==0) {d=a,x=1,y=0;}
```

```
    else{
        gcd(b,a%b,d,y,x);
        y-=x*(a/b);     //关键在于理解此句.
    }
}
```

关于 `y-=x*(a/b)` 的解释： 已知正整数 a,b 满足 $\gcd(a,b)=d$，且整数 x_0,y_0 满足 $ax_0+by_0=d$. 设 $a/b=p,a\%b=r$，这里"/"按计算机中的整除理解. 则有 $(bp+r)x_0+by_0=d$，整理得

$$rx_0+b(px_0+y_0)=d.$$

当从下一层递归中回到上一层时，传回的参数为 $x=x_0,y=px_0+y_0$. 而在上一层递归中我们想得到 $x=x_0,y=y_0$，因此需要 `y-=x*(a/b)`.

应用：求乘法逆元

(1) 乘法逆元：若 $ab\equiv 1 \pmod p$，则 a,b 互为模 p 的乘法逆元.

(2) 在扩展欧几里得算法中取 $b=p$，得到的 x 即为 a 模 p 的乘法逆元.

5 三分法

问题： 求单峰函数的最值.

思想： 以求先增后减的单峰函数 $f(x)$ 在区间 $[l,r]$ 上的极值点，取 x_1,x_2 把区间分成等长的三个子区间 $[l,x_1],[x_1,x_2],[x_2,r]$. 若 $f(x_1)>f(x_2)$，则极值点肯定不在区间 $[x_2,r]$ 上，进而问题变为求 $f(x)$ 在区间 $[l,x_2]$ 上的最值. 由此一步一步缩小范围，最后得到极值点的近似值.

代码：

```
double f(double x);
double solve(double l, double r){
    double x1=l+(r-l)/3;
    double x2=r-(r-l)/3;
    if(fabs(x1-x2)<0.0001) return f(x1);
    if(f(x1)>f(x2)) return solve(l,x2);
    else return solve(x1,r);
}
```

6 排列组合

数学基础：

(1) 排列数 A_m^n：从 m 个互不相同的物品中选出 n 个排成一列，共有 A_m^n 种方式.

(2) 组合数 C_m^n：从 m 个互不相同的物品中选出 n 个，共有 C_m^n 种方式.

(3) 公式：$\mathrm{A}_m^n=\dfrac{m!}{(m-n)!},\mathrm{C}_m^n=\dfrac{m!}{n!(m-n)!}$.

(4) 排列数与组合数的关系：$\mathrm{A}_m^n=\mathrm{C}_m^n\cdot\mathrm{A}_n^n$.

(5) 杨辉三角：$(a+b)^n$ 的各项展开式系数.

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & 1 & & 1 & & & \\
 & & 1 & & 2 & & 1 & & \\
 & 1 & & 3 & & 3 & & 1 & \\
1 & & 4 & & 6 & & 4 & & 1
\end{array}$$

性质：从第三行起，两侧为 1，中间的每个数等于它"肩上"的两个数之和.

6.1 Catalan 数

(1) 基本性质：$h_0=1,h_1=1,h_n=h_0h_{n-1}+h_1h_{n-2}+\cdots+h_{n-1}h_0$.

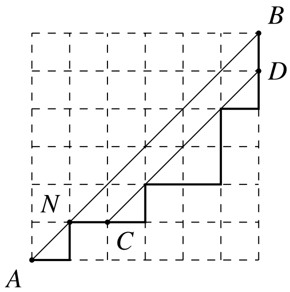
如果能把公式化成上面这种形式，则为 Catalan 数.

(2) 通项：$h_n=\mathrm{C}_{2n}^{n}-\mathrm{C}_{2n}^{n+1}=\dfrac{1}{n+1}\mathrm{C}_{2n}^{n}$.

(3) 证明：折线法.

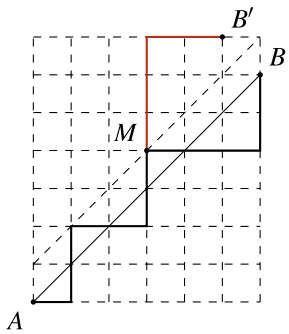
设定如下情境：在平面直角坐标系中，从 $A(0,0)$ 走到 $B(n,n)$，每次可向右走 1 单位或向上走 1 单位. 但在任意时刻，已向右走的步数不少于向上走的步数. 记满足这样条件的路径数为 h_n.

从递推角度考虑. 假设一条合法路径与 $y=x$（图中实线）的最后一个交点（B 之前）为 $N(k,k)$，则 $A-N$ 的合法路径数为 h_k.



从 N 点出发的一步必须向右走，整个路径的最后一步必须向上走. 之间由于路径与 $y=x$ 再无交点，则 $C(k+1,k)-D(n,n-1)$ 可看作是 $n'=n-k-1$ 的一个子问题,因此路径数为 h_{n-k-1}. 所以此时共有 h_kh_{n-k-1} 条路径.

k 的可能取值有 $0,1,\cdots,n-1$，累加得总路径数 $h_n=\sum\limits_{k=0}^{n-1}h_kh_{n-k-1}$.



另一方面，显然我们只能在方格表在 $y=x$ 右下方的部分运动. 不合法的路径将与 $y=x+1$（图中虚线）相交.

对于每条不合法的路径，设其与 $y=x+1$ 的最后一个交点为 M，将 $M-B$ 部分的路线沿 $y=x+1$ 翻折，得到一条到达 $B'(n-1,n+1)$ 的路径.

显然，$A-B$ 的不合法路径与 $A-B'$ 的路径一一对应.

$A-B$ 的路径（含不合法的）共有 C_{2n}^{n} 条，而 $A-B'$ 的路径共有 C_{2n}^{n+1} 条，得 $h_n=\mathrm{C}_{2n}^{n}-\mathrm{C}_{2n}^{n+1}=\dfrac{1}{n+1}\mathrm{C}_{2n}^{n}$.

(4) 应用：

- 出栈次序：一个栈，现在要求将 $1,2,\cdots,n$ 依次入栈，求有多少种不同的出栈顺序.

同证明中的模型对应，入栈即为向右走 1 单位，出栈即为向上走 1 单位，栈中元素不少于 0 个即为任意时刻向右走的步数不少于向上走的步数.

- 二叉树构成问题：一个 n 个顶点的二叉树（节点无编号，但左右叶子不可颠倒）共有多少种？

考虑根节点的左右子树，若根的左子树有 k 个节点，则右子树有 $n-k-1$ 个节点，$k=0,1,\cdots,n-1$，显然为卡特兰数模型.

6.2 Lucas 定理

目的: 大组合数取模.

基本形式: $\mathrm{C}_n^m \equiv \mathrm{C}_{n/p}^{m/p}\mathrm{C}_{n\%p}^{m\%p} \pmod p$, p 必须为素数.

证明: 注意到对任意 $0<f<p$, 有 $p\mathrm{C}_{p-1}^{f-1}=f\mathrm{C}_p^f$, 而 p 为素数, 故 $p \mid \mathrm{C}_p^f$.

设 $n=sp+t, m=qp+r$, 其中 $0<t,r<p$.

则由二项式定理

$$(1+x)^n=\left[(1+x)^p\right]^s\cdot(1+x)^t\equiv\left(1+x^p\right)^s\cdot(1+x)^t \pmod p,$$

注意到上式中, "$\equiv$"左右两个多项式只相差一些系数被 p 整除的项, 因此各项系数模 p 同余.

比较 m 次项系数, 得 $\mathrm{C}_n^m \equiv \mathrm{C}_s^q\mathrm{C}_t^r \pmod p$, 即 $\mathrm{C}_n^m \equiv \mathrm{C}_{n/p}^{m/p}\mathrm{C}_{n\%p}^{m\%p} \pmod p$.

递归: `Lucas(n,m)=C(n%p,m%p)*Lucas(n/p,m/p)%p`.

递归出口 `if(m==0) return 1;`.

代码:

```
int fac[N];
fac[1]=1;
for(int i=2; i<=N; i++) fac[i]=fac[i-1]*i%p;
long long pow(long long a,long long b){
    long long ans=1;
    while(b){
        if(b&1) ans=ans*a%p;
        b>>=1;
        a=a*a%p;
    }
    return ans;
}
long long C(long long n,long long m){
    if(m>n) return 0;
    return fac[n]*pow(fac[m]*fac[n-m],p-2)%p;
}
long long Lucas(long long n,long long m){
    if(m==0) return 1;
    return (C(n%p,m%p)*Lucas(n/p,m/p))%p;
}
```

时间复杂度: $O(p\log_p m)$.

7 矩阵乘法

数学基础:

(1) 矩阵: 一个按照长方阵列排列的复数或实数集合.

(2) 矩阵乘法: 设 A 为 $m\times p$ 的矩阵, B 为 $p\times n$ 的矩阵, 那么称 $m\times n$ 的矩阵 C 为矩阵 A 与 B 的乘积, 记作 $C=AB$, 其中矩阵 C 中的第 i 行第 j 列元素可以表示为

$$(AB)_{ij}=\sum_{k=1}^{p}a_{ik}b_{kj}=a_{i1}b_{1j}+a_{i2}b_{2j}+\cdots+a_{ip}b_{pj}.$$

例：设 $A=\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$，$B=\begin{bmatrix}1&4\\2&5\\3&6\end{bmatrix}$. 则

$$C=AB=\begin{bmatrix}1\times1+2\times2+3\times3 & 1\times4+2\times5+3\times6\\4\times1+5\times2+6\times3 & 4\times4+5\times5+6\times6\end{bmatrix}=\begin{bmatrix}14&32\\32&77\end{bmatrix}.$$

(3) 矩阵乘法的性质：

- 乘法结合律：$(AB)C=A(BC)$；
- 乘法左分配律：$(A+B)C=AC+BC$；
- 乘法右分配律：$C(A+B)=CA+CB$；
- 对数乘的结合性：$k(AB)=(kA)B=A(kB)$；
- 矩阵乘法一般**不满足交换律**.

应用：矩阵快速幂提高递推效率.

例：求斐波那契数列第 n 项 $(a_1=a_2=1, a_{n+2}=a_{n+1}+a_n)$.

$$\begin{bmatrix}a_{n+1}&a_n\end{bmatrix}\begin{bmatrix}1&1\\1&0\end{bmatrix}=\begin{bmatrix}a_{n+1}+a_n&a_{n+1}\end{bmatrix}=\begin{bmatrix}a_{n+2}&a_{n+1}\end{bmatrix}.$$

可以先求出 $n-2$ 个 $\begin{bmatrix}1&1\\1&0\end{bmatrix}$ 连乘的结果，再与 $\begin{bmatrix}1&1\end{bmatrix}$ 相乘即得到 $\begin{bmatrix}a_n&a_{n-1}\end{bmatrix}$.

可以把 $O(n)$ 的递推降到 $O(\log n)$.

8 高斯消元

目的：解多元一次方程组.

过程：以解三元一次方程组 $\begin{cases}2x+y+z=1,\\6x+2y+z=-1,\\-2x+2+z=7\end{cases}$ 为例.

以上方程组可记为 $\begin{bmatrix}x&y&z&val\\2&1&1&1\\6&2&1&-1\\-2&2&1&7\end{bmatrix}$.

用第一行消掉后两行的 x 项 $\begin{bmatrix}x&y&z&val\\2&1&1&1\\0&-1&-2&-4\\0&3&2&8\end{bmatrix}$.

再用第二行消掉第三行的 y 项 $\begin{bmatrix}x&y&z&val\\2&1&1&1\\0&-1&-2&-4\\0&0&-4&-4\end{bmatrix}$.

进而可从第三行得到 $z=1$，从下至上回代，依次得到 $y=2, x=-1$.

注意事项：

(1) 系数不一定是整数——用 double；

(2) 无解情况：某一行系数均为 0，但 val 不为 0；

(3) 多解情况：存在某一行系数和 val 均为 0.