ACM 模板

dnvtmf

2015

目录

1　数据结构

2　动态规划

3　图论

4 数学专题

4.1 逆元

```
///逆元inverse
//定义：如果a * b = 1 (% MOD), 则b 是a的逆元（模逆元， 乘法逆元）
//a的逆元存在条件：gcd(a, MOD) == 1
//性质：逆元是积性函数，如果c = a * b, 则 inv[c] = inv[a] * inv[b] % MOD
//方法一： 循环找解法（暴力）
//O(n)  预处理inv[1-n]：O（n^2)
LL getInv(LL x, LL MOD)
{
    for(LL i = 1; i < MOD; i++)
        if(x * i % MOD == 1)
            return i;
    return -1;
}

//方法二： 费马小定理和欧拉定理
//费马小定理:$a^{(p-1)} \equiv 1(\%p)$，其中p是质数，所以a的逆元是$a^{(p-2)}\%p$
//欧拉定理:$x^{\phi(m)} \equiv 1(\%m)$ x与m互素，m是任意整数
//O(log n)(配合快速幂)，预处理inv[1-n]：O（nlog n)
LL qpow(LL x, LL k, LL MOD){....}
LL getInv(LL x, LL MOD)
{
    //return qpow(x, euler_phi(MOD) - 1, MOD);
    return qpow(x, MOD - 2, MOD);//MOD是质数
}

//方法三：扩展欧几里得算法
//扩展欧几里得算法可解决  $a*x+b*y=gcd(a,b)$
//所以a * x % MOD = gcd(a, b) % MOD (b = MOD)
//O(log n)，预处理inv[1-n]：O(nlog n)
LL exgcd(LL a, LL b, LL &x, LL &y){
    if(b == 0)
    {
        x = 1;
        y = 0;
        return a;
    }
    LL g = exgcd(b, a%b, x, y);
    LL t = x;
    x = y;
    y = t - a / b * y;
    return g;
}
LL getInv(LL x, LL MOD){
    LL inv, y;
    exgcd(x, MOD, inv, y);
    inv = (inv%MOD + MOD)%MOD;
    return inv;
}

//方法四：积性函数
//已处理inv[1] -- inv[n - 1]，求inv[n]， （MOD > n)
//MOD = x*n - y(0 <= y < n), ==> x * n = y (% MOD), ==> x * n * inv[y] = y * inv[y] = 1 (%MOD)
//所以inv[n] = x * inv[y] (x = MOD - MOD / n, y  MOD % n)
//O(log n) 预处理inv[1-n]：O(n)
LL inv[NUM];
void inv_pre(LL mod)
{
    inv[0] = inv[1] = 1LL;
    for(int i = 2; i < NUM; i++)
```

```
        inv[i] = (mod - mod/i)*inv[mod%i] % mod;
}

//方法五：积性函数+因式分解
//预处理出所有质数的的逆元，采用exgcd来实现素数O(log n)求逆
//采用质因数分解， 可在O(log n)求出任意一个数的逆元
//预处理O(n log n), 单个O(log n)
```

4.2 模

```
/*
 模(Module)
 1. 基本运算
    Add: (a + b) % p = (a % p + b % p) % p
    Subtract:(a - b) % p = ((a%p - b%p)%p + p)%p
    Multiply:(a * b) % p = ((a % p) * (b % p)) % p
    Dvidive: (a / b) % p = (a * b^{-1}) % p b^{-1}是b关于p的逆元
    Power: (a^b)%p = ((a%p)^b)%p
 2. 推论
    若a ≡ b(%p), c ≡ d(%p),则(a + c) ≡ (b + d)(%p), (a − c) ≡ (b − d)(%p), (a ∗ c) ≡ (b ∗ d)(%p), (a/c) ≡ (b/d)(%p)

 3. 费马小定理
    若p是素数，对任意正整数x,有 x^p ≡ x(%p).
 4. 欧拉定理
    若p与x互素，则有x^{φ(p)} ≡ 1(%p).
 5. n!
    p是素数，n! = a p^e, e = (n/p + n/p^2 + n/p^3 + ···)(a不能被p整除)
    威尔逊定理：(p-1)! ≡ -1 (%p)
    n!中不能被p整除的数的积:n! = (p − 1)!^{(n/p)} × (n mod p)!
    n!中能被p整除的项为:p, 2p, 3p, ... , (n/p)p, 除以p得到1,2,3,...,n/p(问题从缩减到n/p)
    在O(p)时间内预处理除0 <= n < p范围内中的n! mod p的表
    可在O(log_p n)时间内算出答案
    若不预处理,复杂度为O(p log_p n)
 */
int fact[MAX_P];//预处理n! mod p的表.O(p)
//分解n! = a p^e.返回a % p. O(log_p n)
int mod_fact(int n, int p, int &e)
{
    e = 0;
    if(n == 0) return 1;
    //计算p的倍数的部分
    int res = mod_fact(n / p, p, e);
    e += n/p;
    //由于(p − 1)! ≡ −1,因此只需知n/p的奇偶性
    if(n / p % 2) return res * (p - fact[n%p]) % p;
    return res * fact[n % p] % p;
}
```

4.3 中国剩余定理和线性同余方程组

```
/*线性同余方程
a_i × x ≡ b_i(mod m_i) (1 ≤ i ≤ n)
如果方程组有解，那么一定有无穷有无穷多解，解的全集可写为x ≡ b(mod m)的形式.
对方程逐一求解.令b = 0, m = 1;
1.x ≡ b(mod m)可写为x = b + m * t;
2. 带入第i个式子：a_i(b + m ∗ t) ≡ b_i(mod m_i),即a × m × t ≡ b_i − a_i × b_i(mod m_i)
3.当gcd(m_i, a_i × m)无法整除b_i − a_i × b时原方程组无解
*/
```

4.4 组合与组合恒等式

/*1. 组合：从n个不同的元素中取r个的方案数C_n^r：

$$C_n^r=\begin{cases}\dfrac{n!}{r!(n-r)!}, & n\ge r\\ 1, & n\ge r=0\\ 0, & n<r\end{cases}$$

推论1：$C_n^r=C_n^{n-r}$

推论2(Pascal公式)：$C_n^r=C_{n-1}^r+C_{n-1}^{r-1}$

推论3：$\sum_{k=r-1}^{n-1}C_k^{r-1}=C_{n-1}^{r-1}+C_{n-2}^{r-2}+\ldots+C_{r-1}^{r-1}=C_n^r$

2. 从重集$B=\{\infty\cdot b_1,\infty\cdot b_2,\ldots,\infty\cdot b_n\}$的r−组合数F(n,r)为

$F(n,r)=C_{n+r-1}^r$

3. 二项式定义

当n是一个正整数时，对任何x和y有：

$$(x+y)^n=\sum_{k=0}^n C_n^k x^k y^{n-k}$$

令y=1，有：

$(1+x)^n=\sum_{k=0}^n C_n^k x^k=\sum_{k=0}^n C_n^{n-k}x^k$

广义二项式定理：

广义二项式系数：对于任何实数α和整数k,有

$$C_\alpha^K=\begin{cases}\frac{\alpha(\alpha-1)\ldots(\alpha-k+1)}{k!} & k>0\\ 1 & k=0\\ 0 & k<0\end{cases}$$

设α是一个任意实数，则对满足$|\frac{x}{y}|<1$的所有x和y，有

$$(x+y)^\alpha=\sum_{k=0}^\infty C_\alpha^k x^k y^{\alpha-k}$$

推论：令$z=\frac{x}{y}$,则有

$$(1+z)^\alpha=\sum_{k=0}^\infty C_\alpha^k z^k,|z|<1$$

令$\alpha=-n$(n是正整数),有

$$(1+z)^{-n}=\frac{1}{(1+z)^n}=\sum_{k=0}^\infty(-1)^k C_{n+k-1}^k z^k$$

又令z = −rz，(r为非零常数),有

又令n=1,有

$$\frac{1}{1+z}=\sum_{k=0}^\infty(-1)^k z^k$$

令z = −z，有

$$\frac{1}{1-z}=\sum_{k=0}^\infty z^k$$

令$\alpha=\frac{1}{2}$，有

$$\sqrt{1+z}=1+\sum_{k=1}^\infty\frac{(-1)^{k-1}}{k\cdot 2^{2k-1}}C_{2k-2}^{k-1}z^k$$

4. 组合恒等式

1. $\sum_{k=0}^n C_n^k=2^n$
2. $\sum_{k=0}^n(-1)^k C_n^k=0$
3. 对于正整数 n 和 k，

$$C_n^k=\frac{n}{k}C_{n-1}^{k-1}$$

4. 对于正整数 n，

$$\sum_{k=0}^{n} kC_n^k = \sum_{k=1}^{n} kC_n^k = n \cdot 2^{n-1}$$

5. 对于正整数 n，

$$\sum_{k=0}^{n} (-1)^k kC_n^k = 0$$

6. 对于正整数 n，

$$\sum_{k=0}^{n} k^2 C_n^k = n(n+1)2^{n-2}$$

7. 对于正整数 n，

$$\sum_{k=0}^{n} \frac{1}{k+1} C_n^k = \frac{2^{n+1}-1}{n+1}$$

8. (Vandermonde 恒等式) 对于正整数 n,m 和 p，有 $p \leq minm, n$，

$$\sum_{k=0}^{p} C_n^k C_m^{p-k} = C_{m+n}^p$$

9. (令 p=m) 对于任何正整数 n,m，

$$\sum_{k=0}^{m} C_m^k C_n^k = C_{m+n}^m$$

10. (又令 m=n) 对于任何正整数 n，

$$\sum_{k=0}^{n} (C_n^k)^2 = C_{2n}^n$$

11. 对于非负整数 p,q 和 n，

$$\sum_{k=0}^{p} C_p^k C_q^k C_{n+k}^{p+q} = C_n^p C_n^q$$

12. 对于非负整数 p,q 和 n，

$$\sum_{k=0}^{p} C_p^k C_q^k C_{n+p+q-k}^{p+q} = C_{n+p}^p C_{n+q}^q$$

13. 对于非负整数 n,k，

$$\sum_{i=0}^{n} C_i^k = C_{n+1}^{k+1}$$

14. 对于所有实数 α 和非负整数 k，

$$\sum_{j=0}^{k} C_{\alpha+j}^j = C_{\alpha+k+1}^k$$

15.

$$\sum_{k=0}^{n} \frac{2^{k+1}}{k+1} C_n^K = \frac{3^{n+1}-1}{n+1}$$

16.

$$\sum_{k=0}^{m} C_{n-k}^{m-k} = C_{n+1}^m$$

17.

$$\sum_{k=m}^{n} C_k^m C_n^k = C_n^m 2^{n-m}$$

18.

$$\sum_{k=0}^{m} (-1)^k C_n^k = (-1)^m C_{n-1}^m$$

*/

5　字符串

6 Java

```
import java.io.*;
import java.util.*;
import java.math.*;
import java.BigInteger;

public class Main{
    public static void main(String arg[]) throws Exception{
        Scanner cin = new Scnner(System.in);

        BigInteger a, b;
        a = new BigInteger(”123”);
        a = cin.nextBigInteger();
        a.add(b);//a + b
        a.subtract(b);// a – b
        a.multiply(b);// a * b
        a.divide(b);// a / b
        a.negate();// –a
        a.remainder(b);//a%b
        a.abs();//|a|
        a.pow(b);//a^b
        //.... and other math fuction, like log();
        a.toString();
        a.compareTo(b);//

    }
}
```