ACM 模板

dnvtmf

2015

目录

1 数据结构 3

2 动态规划 4

3 图论 5

4 数学专题 6

4.1 逆元 6

5 字符串 8

6 Java 9

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a^(p−1) = 1(%p)，其中p是质数，所以a的逆元是a^(p−2)%p
//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, MOD — 2, MOD);
}

//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)
```

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);//

    }
}
```