Maths for Programming Contests

Basics

- Number Represenation in varius bases
- Basic operations
- Divisibility
- Reading long long long numbers
- Basic math functions like factorial

http://www.spoj.pl/problems/FCTRL/

GCD of Two Numbers

- If b|a then gcd(a,b) = b
- Otherwise a = bt+r for some t and r
- gcd(a,b) = gcd(b,r)
- gcd(a,b) = gcd(b,a%b)
- Lcm(a,b) = (a*b)/gcd(a,b)

http://www.spoj.pl/problems/GCD2/

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

Prime Numbers

- Definition
- Checking if a number is prime
- Generating all prime factors of a number
- Generating all prime numbers less than a given number (**Sieve of Eratosthenes)**

http://www.codechef.com/problems/PRIMES2/

Modular Arithmetic

- $(x+y) \bmod n = ((x \bmod n) + (y \bmod n)) \bmod n$
- $(x-y) \bmod n = ((x \bmod n) - (y \bmod n)) \bmod n$
- $(x*y) \bmod n = (x \bmod n)(y \bmod n) \bmod n$
- ~~$(x/y) \bmod n = ((x \bmod n)/(y \bmod n)) \bmod n$~~
- $(x^y) \bmod n = (x \bmod n)^y \bmod n$
- $(x^y) \bmod n=((x^{y/2}) \bmod n)\ ((x^{y/2}) \bmod n) \bmod n$
- a simple recursive function to compute the value in O(log n) time.

Modular Exponentiation

- Another way to compute it in O(log n) time
- $Y = a_0 + a_1*2 + a_2*2^2+\ldots+a_k*2^k$
- $X^{(a+b)} = X^a*X^b$ $\qquad X^{ab} = (X^a)^b$
- $X^Y = X^{a0} + (X^2)^{a1} + (X^{2\char`^2})^{a2} +\ldots.+(X^{2\char`^k})^{ak}$

```
int exp(int x,int y,int mod){
        int result=1;
        while(y){
                if((y & 1) == 1){
                        result = (result*x)%mod;
                }
                y = y >> 1;
                x = (x*x)%mod;
        }
        return result;
}
```

Modular Multiplication

```
int mulmod(int a,int b,int MOD){
    int result = 0,tmp=a,x=b;
    while(x){
        if(x&1){
            result += tmp;
            if(result >=MOD)
                result -= MOD;
        }
        tmp += tmp;
        if(tmp >= MOD)
            tmp -= MOD;
        x = x>>1;
    }
    return result;
}
```

Euler's totient function

- Φ(n) – Number of positive integers less than or equal to n which are coprime to n
- $\Phi(ab) = \phi(a)\ \phi(b)$
- For a prime p $\phi(p) = p-1$
- For a prime p $\phi(p^k) = p^k - p^{k-1} = p^k(1-1/p)$
- $N = (p_1^{k1})*(p_2^{k2})*\ldots*(p_n^{kn})$
- $\Phi(N) = (p_1^{k1}(1-1/p_1))*\ldots*(p_n^{kn}(1-1/p_n))$
- $\Phi(N) = N(1-1/p_1)(1-1/p_2)\ldots..(1-1/p_n)$

Sieve technique for Euler's function

```
void prefill(){
    int i,j;
    phi[1] = 0;
    for(i=2;i<=MAX;i++){
        phi[i] = i;
    }
    for(i=2;i<=MAX;i++){
        if(phi[i] == i){
            for(j=i;j<=MAX;j+=i){
                phi[j] = (phi[j]/i)*(i-1);
            }
        }
    }
}
```

Euler’s Theorem

- When a and n are co-prime
- if $x \equiv y \pmod{\phi(n)}$, then $a^x \equiv a^y \pmod{n}$.
- $a^{\phi(n)} \equiv 1 \pmod{n}$ (actual theorem the above is a generalization)
- Euler’s Theorem is a genaralization for Fermat's little theorem

Counting

- Product Rule
- Sum Rule
- Principle of inclusion and exclusion
- Closed form expression
- Recurrence Relations
- Use of DP to compute the values using the above recurren relations

Additional things to look up

- Extended Euclidean algorithm for GCD
- Chinese Remaindering theorem
- Mobius Function and its computation using seive

Practice

- http://www.spoj.pl/problems/AU12/
- http://www.spoj.pl/problems/PRIME1/
- http://www.spoj.pl/problems/LASTDIG2/
- http://www.spoj.pl/problems/AUCSE015/
- http://www.spoj.pl/problems/MAIN111/
- http://www.spoj.pl/problems/PRIMES2/
- http://www.spoj.pl/problems/IITD4/
- http://www.spoj.pl/problems/NDIVPHI/
- http://www.spoj.pl/problems/TRICOUNT/
- http://www.spoj.pl/problems/CUBEFR/
- http://www.spoj.pl/problems/NOSQ/
- http://www.spoj.pl/problems/MMOD29/
- http://www.spoj.pl/problems/PROGPROG/