Number Theory
(ทฤษฎีจำนวน)

Number Theory

- Prime Numbers
- Greatest Common Divisor & Least Common Multiple
- Factorial
- Finding Prime Factors with Optimized Trial Divisions
- Modified Sieve
- Modulo Arithmetic
- Extended Euclid

การหารลงตัว

- If x and y are integers with $x \neq 0$, we say that x *divides* y if there is an integer c such that $y = xc$, or equivalently, if $\frac{y}{x}$ is an integer.
- When x divides y we say that x is a *factor* or *divisor* of y, and that y is a *multiple* of x.
- The notation $x \mid y$ denotes that x divides y.
- We write $x \nmid y$ when x does **not** divide y.

ตัวอย่าง การหารลงตัว

- จงพิจารณาว่า $3 \mid 7$ และ $3 \mid 12$ หรือไม่
- Let *n* and *d* be positive integers.
  How many positive integers NOT exceeding *n* are divisible by *d*?

THEOREM 1

- Let a, b, and c be integers, where $a \neq 0$. Then
- (i) if $a \mid b$ and $a \mid c$, then $a \mid (b + c)$
- (ii) if $a \mid b$, then $a \mid bc$ for all integers c;
- (iii) if $a \mid b$ and $b \mid c$, then $a \mid c$.

COROLLARY 1

- If a, b, and c are integers, where $a = 0$, such that $a \mid b$ and $a \mid c$, then $a \mid mb + nc$ whenever m and n are integers.

THEOREM 2

- **THE DIVISION ALGORITHM**
- Let a be an integer and d a positive integer.
- Then there are unique integers q and r, with $0 \leq r < d$, such that $a = d\,q + r$.
- $q = a$ **div** d,
- $r = a$ **mod** d.

ตัวอย่าง

- What are the quotient(ผลลัพธ์) and remainder (เศษ) when 101 is divided by 11?
- What are the quotient and remainder when −11 is divided by 3?

Modular Arithmetic

- If a and b are integers and m is a positive integer, then a is *congruent to b modulo m* if $m \mid (a - b)$.
- $a \equiv b \pmod{m}$ to indicate that a is congruent to b modulo m.
- $a \equiv b \pmod{m}$ is a **congruence** and that m is its **modulus**.
- If a and b are **NOT** congruent modulo m, we write $a \not\equiv b \pmod{m}$.

- จงพิจารณาว่า 17 is congruent to 5 modulo 6
- และ 24 and 14 are congruent modulo 6 หรือไม่

THEOREM 3

- Let a and b be integers,
  and let m be a positive integer.
- Then $a \equiv b \pmod{m}$ **if and only if**
  $a \ \mathbf{mod} \ m = b \ \mathbf{mod} \ m$.

THEOREM 4

- Let m be a positive integer.
- The integers a and b are **congruent modulo** m **if and only if** there is an integer k such that $a = b + km$.

THEOREM 5

- Let m be a positive integer.
- If $a \equiv b \pmod{m}$ and $c \equiv d \pmod{m}$,
- Then $a + c \equiv b + d \pmod{m}$
- and $ac \equiv bd \pmod{m}$.

COROLLARY 2

- Let m be a positive integer and let a and b be integers.
  Then
  $$(a + b) \mathbf{\ mod\ } m = ((a \mathbf{\ mod\ } m) + (b \mathbf{\ mod\ } m)) \mathbf{\ mod\ } m$$

and

$$ab \mathbf{\ mod\ } m = ((a \mathbf{\ mod\ } m)(b \mathbf{\ mod\ } m)) \mathbf{\ mod\ } m.$$

จำนวนเต็ม

- **THEOREM 1**
- Let b be an integer greater than 1.
- If n is a positive integer,
  it can be expressed uniquely in the form
- $n = a_k b^k + a_{k-1} b^{k-1} + \cdot \quad \cdot \quad \cdot + a_1 b + a_0,$
- where k is a nonnegative integer,
  $a_0, a_1, \ldots, a_k$ are nonnegative integers less than b,
  and $a_k \neq 0$.

ตัวอย่าง

- จงเปลี่ยน $(1\ 0101\ 1111)_2$ เป็นเลขฐานสิบ
- จงเปลี่ยน 241 เป็นเลขฐานสอง
- จงเปลี่ยน $(7016)_8$ เป็นเลขฐานสิบ
- จงเปลี่ยน 12345 เป็นเลขฐานแปด
- จงเปลี่ยน $(2AE0B)_{16}$ เป็นเลขฐานสิบ
- จงเปลี่ยน 177130 เป็นเลขฐานสิบหก
- จงเปลี่ยน $(7016)_8$ เป็นเลขฐานสอง
- จงเปลี่ยน $(7016)_8$ เป็นเลขฐานสิบหก
- จงเปลี่ยน $(11\ 1110\ 1011\ 1100)_2$ เป็นเลขฐานแปดและเลขฐานสิบหก
- จงเปลี่ยน $(765)_8$ และ $(A8D)_{16}$ เป็นเลขฐานสอง

ALGORITHM 1 Constructing Base *b* Expansions

procedure *base b expansion*(n, b: positive integers with $b > 1$)

$q := n$

$k := 0$

while ($q \neq 0$) {

$a_k := q$ **mod** b

$q := q$ **div** b

$k := k + 1$

}

return $(a_{k-1}, \ldots, a_1, a_0)$

$\{(a_{k-1}, \ldots, a_1, a_0)_b$ is the base b expansion of $n\}$

Integer Operations

- จงหาผลบวกของ $(1110)_2$ กับ $(1011)_2$.

ALGORITHM 2 Addition of Integers.

procedure *add*(a, b: positive integers)
{the binary expansions of a and b are $(a_{n-1}a_{n-2}\ldots a_1a_0)_2$
and $(b_{n-1}b_{n-2}\ldots b_1b_0)_2$, respectively}
$c := 0$
for $j := 0$ **to** $n - 1$
    $d := \lfloor (a_j + b_j + c)/2 \rfloor$
    $s_j := a_j + b_j + c - 2d$
    $c := d$
$s_n := c$
return $(s_0, s_1, \ldots, s_n)$ {the binary expansion of the sum is $(s_ns_{n-1}\ldots s_0)_2$}

Integer Operations

- จงหาผลคูณของ $(110)_2$ กับ $(101)_2$.

ALGORITHM 3 Multiplication of Integers.

procedure *multiply*(a, b: positive integers)
{the binary expansions of a and b are $(a_{n-1}a_{n-2}\ldots a_1a_0)_2$
 and $(b_{n-1}b_{n-2}\ldots b_1b_0)_2$, respectively}
for $j := 0$ **to** $n - 1$
  if $b_j = 1$ **then** $c_j := a$ shifted j places
  else $c_j := 0$
{$c_0, c_1, \ldots, c_{n-1}$ are the partial products}
$p := 0$
for $j := 0$ **to** $n - 1$
  $p := p + c_j$
return p {p is the value of ab}

ALGORITHM 4 Computing div and mod.

procedure *division algorithm*(a: integer, d: positive integer)
$q := 0$
$r := |a|$
while $r \geq d$
    $r := r - d$
    $q := q + 1$
if $a < 0$ and $r > 0$ **then**
    $r := d - r$
    $q := -(q + 1)$
return (q, r) {$q = a$ **div** d is the quotient, $r = a$ **mod** d is the remainder}

ALGORITHM 5 Modular Exponentiation.

procedure *modular exponentiation*(b: integer, $n = (a_{k-1}a_{k-2}\ldots a_1a_0)_2$,
    m: positive integers)
$x := 1$
$power := b$ **mod** m
for $i := 0$ **to** $k - 1$
    if $a_i = 1$ **then** $x := (x \cdot power)$ **mod** m
    $power := (power \cdot power)$ **mod** m
return x {x equals b^n **mod** m}

- จงหา 2^{644} **mod** 645 โดยใช้ algorithm 5
- จงหาเศษที่ได้จากการหาร $13^{401} + 5$ ด้วย 10

Prove that $\forall n \in \underbrace{N, n^2 + n + 41 \text{ is a prime number}}_{\text{Predicate}}$

$N = \{0,1,2,3, \ldots\}$ *Natural number*

จำนวนเฉพาะ

- An integer $p > 1$ is called ***prime*** if the **only** positive factors of p are 1 and p.
- A positive integer > 1 and is **not prime** is called ***composite***.(จำนวนประกอบ)
- **Example**
- The integer 7 is **prime**

  because its **only** positive factors are 1 and 7
- The integer 9 is **composite**
  because it is divisible by 3.

Table of Prime number

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Prime Testing

- test if N is divisible by *divisor* $\in [2..N\text{-}1]$
  $O(N)$
- test if N is divisible by a *divisor* $\in [2..\sqrt{n}]$
  $O(\sqrt{n})$
- test if N is divisible by *divisor* $\in [3, 5, 7, .., \sqrt{n}]$
  $O(\sqrt{n}/2)$
- Test if N is divisible by *prime divisors* $\leq \sqrt{n}]$
  $O(\sqrt{n}/\ln(\sqrt{n}))$

Sieve of Eratosthenes

- generate a list of prime numbers between range [0..N]
- First, sets all numbers in the range to be ‘probably prime’ except 0,1
- Then, it takes 2 as prime and crosses out all multiples of 2 starting from $2\times 2 = 4, 6, . . \leq N$.
- Do the same for 3, 5, 7, … prime#
- $O(N \log \log N)$

Sieve of Eratosthenes

```
#include <bitset>      // compact STL for Sieve, better than vector<bool>!
ll _sieve_size;                 // ll is defined as: typedef long long ll;
bitset<10000010> bs;                // 10^7 should be enough for most cases
vi primes;                  // compact list of primes in form of vector<int>

void sieve(ll upperbound) {      // create list of primes in [0..upperbound]
  _sieve_size = upperbound + 1;              // add 1 to include upperbound
  bs.set();                                          // set all bits to 1
  bs[0] = bs[1] = 0;                               // except index 0 and 1
  for (ll i = 2; i <= _sieve_size; i++) if (bs[i]) {
    // cross out multiples of i starting from i * i!
    for (ll j = i * i; j <= _sieve_size; j += i) bs[j] = 0;
    primes.push_back((int)i);       // add this prime to the list of primes
} }                                          // call this method in main method

bool isPrime(ll N) {            // a good enough deterministic prime tester
  if (N <= _sieve_size) return bs[N];                // O(1) for small primes
  for (int i = 0; i < (int)primes.size(); i++)
    if (N % primes[i] == 0) return false;
  return true;                  // it takes longer time if N is a large prime!
}                    // note: only work for N <= (last prime in vi "primes")^2
```

```
// inside int main()
  sieve(10000000);                      // can go up to 10^7 (need few seconds)
  printf("%d\n", isPrime(2147483647));                     // 10-digits prime
  printf("%d\n", isPrime(136117223861LL));    // not a prime, 104729*1299709
```

Source code: ch5_06_primes.cpp/java

Greatest Common Divisors

- Let a and b be integers, not both zero.
- The largest integer d such that $d \mid a$ and $d \mid b$ is called the *greatest common divisor* of a and b.
- The greatest common divisor of a and b is denoted by $\gcd(a, b)$.
- The integers a and b are *relatively prime* if their greatest common divisor is 1.

- What is the greatest common divisor of
  17 and 22?
  24 and 36?
  120 and 500?

The Euclidean Algorithm

- Let $a = bq + r$,
  where a, b, q, and r are integers.
- Then $\gcd(a, b) = \gcd(b, r)$.

ALGORITHM 1 The Euclidean Algorithm.

procedure $gcd(a, b$: positive integers)
$x := a$
$y := b$
while $y \neq 0$
  $r := x$ **mod** y
  $x := y$
  $y := r$
return $x\{\gcd(a, b)$ is $x\}$

least common multiple

- The *least common multiple* of the positive integers a and b
  is the smallest positive integer that
  is divisible by both a and b.

The least common multiple of a and b is
lcm$(a, b) = a \times b/gcd(a, b)$.

- What is the least common multiple of $2^3 3^5 7^2$ and $2^4 3^3$?

THEOREM 1

- **THE FUNDAMENTAL THEOREM OF ARITHMETIC**
- Every integer greater than 1
  can be written uniquely as a prime
  or as the **product** of two or more primes
- where the prime factors are written in order of nondecreasing size.

ตัวอย่าง

- จงหา prime factorizations ของ 100, 101, 641, 999, และ 1024

THEOREM 2

- If n is a **composite** integer,
  then n has a prime divisor less than or equal to $\sqrt{n}$
- ตัวอย่าง

  - จงแสดงว่า 103 เป็นจำนวนเฉพาะหรือไม่
  - จงแสดงว่า 2873 เป็นจำนวนเฉพาะหรือไม่
  - จงแสดงว่า 7007 เป็นจำนวนเฉพาะหรือไม่

```
vi primeFactors(ll N) {     // remember: vi is vector<int>, ll is long long
  vi factors;
  ll PF_idx = 0, PF = primes[PF_idx]; // primes has been populated by sieve
  while (PF * PF <= N) {              // stop at sqrt(N); N can get smaller
    while (N % PF == 0) { N /= PF; factors.push_back(PF); }    // remove PF
    PF = primes[++PF_idx];                           // only consider primes!
  }
  if (N != 1) factors.push_back(N);           // special case if N is a prime
  return factors;      // if N does not fit in 32-bit integer and is a prime
}                   // then 'factors' will have to be changed to vector<ll>

// inside int main(), assuming sieve(1000000) has been called before
  vi r = primeFactors(2147483647);         // slowest, 2147483647 is a prime
  for (vi::iterator i = r.begin(); i != r.end(); i++) printf("> %d\n", *i);

  r = primeFactors(136117223861LL);                  // slow, 104729*1299709
  for (vi::iterator i = r.begin(); i != r.end(); i++) printf("# %d\n", *i);

  r = primeFactors(142391208960LL);         // faster, 2^10*3^4*5*7^4*11*13
  for (vi::iterator i = r.begin(); i != r.end(); i++) printf("! %d\n", *i);
```

Functions involving with prime

numPF(N): Count the number of *prime factors* of N

```
ll numPF(ll N) {
  ll PF_idx = 0, PF = primes[PF_idx], ans = 0;
  while (PF * PF <= N) {
    while (N % PF == 0) { N /= PF; ans++; }
    PF = primes[++PF_idx];
  }
  if (N != 1) ans++;
  return ans;
}
```

- numDiv(N): Count the number of *divisors* of N
- If a number $N = a^i \times b^j \times ... \times c^k$,
  then N has $(i+1)\times(j+1)\times ... \times(k+1)$ divisors

```
ll numDiv(ll N) {
  ll PF_idx = 0, PF = primes[PF_idx], ans = 1;   // start from ans = 1
  while (PF * PF <= N) {
    ll power = 0;                                 // count the power
    while (N % PF == 0) { N /= PF; power++; }
    ans *= (power + 1);                           // according to the formula
    PF = primes[++PF_idx];
  }
  if (N != 1) ans *= 2;   // (last factor has pow = 1, we add 1 to it)
  return ans;
}
```

Exercises

- **UVa 10140 - Prime Distance ***
  (sieve; linear scan)
- **UVa 10407 - Simple Division ***
  (subtract the set s with s[0], find gcd)
- **UVa 00324 - Factorial Frequencies ***
  (count digits of n! up to 366!)

Linear Congruences

- A congruence of the form
  $ax \equiv b \ (\text{mod } m)$,
- where m is a positive integer,
  a and b are integers, and
  x is a variable,
- is called a **linear congruence**.

- จงหา sequence linear congruence ของ

  $5x \equiv 2 \pmod{6}$

  $2x \equiv 3 \pmod{4}$

  $2x \equiv 6 \pmod{8}$

Inverse of modulo m

- If *a* and *m* are relatively prime integers
  and $m > 1$,
  then an inverse of *a* modulo *m* exists.
- Furthermore, this inverse is unique modulo *m*.
- there is a unique positive integer *a*

  less than *m* that is an inverse of *a* modulo *m*
  and every other inverse of *a* modulo *m* is
- congruent to *a* modulo *m*.)

- the linear congruence $ax \equiv b \pmod{m}$,
- $\overline{a}a \equiv 1 \pmod{m}$,
- $\overline{a}$: an **inverse** of a modulo m

Find an inverse of 101 modulo 4620.

$$\begin{aligned}
4620 &= 45 \cdot 101 + 75 \\
101 &= 1 \cdot 75 + 26 \\
75 &= 2 \cdot 26 + 23 \\
26 &= 1 \cdot 23 + 3 \\
23 &= 7 \cdot 3 + 2 \\
3 &= 1 \cdot 2 + 1 \\
2 &= 2 \cdot 1.
\end{aligned}$$

Gcd(4620,101)

$$\begin{aligned}
1 &= 3 - 1 \cdot 2 \\
&= 3 - 1 \cdot (23 - 7 \cdot 3) = -1 \cdot 23 + 8 \cdot 3 \\
&= -1 \cdot 23 + 8 \cdot (26 - 1 \cdot 23) = 8 \cdot 26 - 9 \cdot 23 \\
&= 8 \cdot 26 - 9 \cdot (75 - 2 \cdot 26) = -9 \cdot 75 + 26 \cdot 26 \\
&= -9 \cdot 75 + 26 \cdot (101 - 1 \cdot 75) = 26 \cdot 101 - 35 \cdot 75 \\
&= 26 \cdot 101 - 35 \cdot (4620 - 45 \cdot 101) = -35 \cdot 4620 + 1601 \cdot 101.
\end{aligned}$$

Inverse of modulo 101

The Chinese Remainder Theorem

- There are certain things whose number is unknown. When divided by 3, the remainder is 2; when divided by 5, the remainder is 3; and when divided by 7, the remainder is 2.
- What will be the number of things?

$x \equiv 2 \pmod{3}$,

$x \equiv 3 \pmod{5}$,

$x \equiv 2 \pmod{7}$?

- Let $m_1, m_2, . . . , m_n$ be pairwise relatively
- prime positive integers greater than one and $a1, a2, . . . , an$ arbitrary integers. Then the system
- $x \equiv a_1$ (mod m_1),
- $x \equiv a2$ (mod $m2$),
- ▪
- ▪
- ▪
- $x \equiv an$ (mod mn)
- has a unique solution modulo $m = m_1m2$ ▪ ▪ ▪ mn. (That is, there is a solution x with
- $0 \le x < m$, and all other solutions are congruent modulo m to this solution.)

- Given any 4 positive integer, some pair of them will have a difference divisible of 3.

True or False ?