# Elementary Number Theory

Mushthofa & Julio Adisantoso

## Keterbagian

- Asumsi : Bilangan = bilangan bulat
- Bilangan a adalah pembagi dari bilangan b, jika dan hanya jika b = qa, untuk suatu bilangan bulat q.
- Notasi: a | b
- Jika a pembagi dari b, maka b adalah kelipatan dari a
- Banyaknya kelipatan k pada himpunan 1 s/d n = [n/k]

# Prima

- Prima: bilangan bulat positif > 1 sedemikian hingga

$$a \mid p \rightarrow (a = 1 \vee a = p)$$

- Jika p prima, maka

$$p \mid ab \rightarrow (p \mid a \vee p \mid b)$$

- p adalah prima → ada tepat dua pembagi dari p

# Sieve of Eratosthenes

- Metode menghasilkan/"generate" bilangan prima pada sebuah range (1 s/d n)
- Pertama, buat daftar bilangan 2 s/d n. Dengan i mulai dari elemen pertama (i=2):
  - Hapus semua kelipatan i mulai dari i*i yang ada di daftar
- Elemen yang tersisa adalah prima
- Efisien sampai dengan n ≈ 10 juta

# Sieve of Eratosthenes

```
bitset<10000005> bs;
vi primes;
void sieve(long long int n)
{
      bs.set();
      bs[0] = bs[1] = 0;
      for(long long int i=2;i<=n+1;i++)
      {
       if(bs[i])
       {
        for(long long int j=i*i; j<=n+1; j+=i) bs[j]=0;
        primes.push_back((int)i);
       }
      }
}
```

# Pengecekan Prima

- Apakah sebuah bilangan bulat a adalah prima?
- Naive: Periksa semua k, k=2 s/d a-1 apakah ada yang k | a
- Better: Cukup periksa dengan $k = 2...\left\lfloor \sqrt{a} \right\rfloor$
  - Karena jika pq = a, maka min(p,q) $\leq \sqrt{a}$
- Even better: (cukup untuk kondisi kontes):
  - Panggil sieve($[\sqrt{a}]$)
  - Periksa untuk setiap prima p $\leq \sqrt{a}$ apakah p|a
- Implementasi?

# Aritmetika Modular

- **Teorema Pembagian**: Untuk setiap bilangan bulat positif a dan b, terdapat unik q dan r bilangan bulat s.r.s.

$$a = bq + r$$
$$0 \le r \le b - 1$$

- q adalah *hasil bagi* a dengan b, q = a div b
- r adalah *sisa bagi* a dengan b, r = a mod b, atau a= r (mod b)
- Contoh 1000 mod 3 = 1 dan 1000 mod 7 = 6
- Jika a mod b = 0 (atau a=0 (mod b)) maka b | a

# GCD

- Greatest Common Divisor (GCD) atau Faktor Persekutuan Terbesar (FPB) dari dua bilangan a dan b adalah bilangan terbesar d sehingga d | a dan d | b
- gcd(p, q) = 1 jika p atau q adalah prima serta salah satu tidak saling membagi
- Jika gcd(a,b) = 1, kita katakan a dan b saling prima (co-prime).

# Metode Euclid

- Metode Iteratif

```
function gcd(a, b)
    while b ≠ 0
       t := b
       b := a mod b
       a := t
    return a
```

- Metode Rekursif

```
function gcd(a, b)
    if b = 0
       return a
    else
       return gcd(b, a mod b)
```

# Bezout's Theorem

- Jika d = gcd(a,b) maka ada x dan y bulat s.r.s,

$$d = ax + by$$

- *z=ax+by*, jika dan hanya jika gcd(a,b) | z
- Contoh:
  - 3 = gcd(12, 21), dan 3 = 12*(-5) + 21*3
  - Semua bilangan yang berbentuk 12x+21y habis dibagi 3
- Tentukan x dan y dalam gcd(178, 312)?

# Problems

- Tentukan semua pasangan bilangan bulat x dan y sedemikian
  1. 15x + 27y = 6 ?
  2. 25x + 40y = 12?
- **(OSP 2012)** : Pak Dengklek memiliki 2 buah takaran air, A dan B, masing-masing volumenya adalah 35 ml dan 48 ml. Jika Pek Dengklek ingin mengambil tepat 22 ml air, maka Pak Dengklek dapat melakukannya dengan menggunakan tiga langkah penakaran, yaitu: takar 2 kali dengan takaran A (2x35=70 ml) lalu kurangkan dengan 1 kali takaran B (70-48=22). Jika Pak Dengklek ingin mengukur tepat 10 ml air, berapakah minimal penakaran yang diperlukan?

# Extended Euclid

- Nilai x dan y dalam Bezout's Identity dapat dihitung dengan Algoritme Extended Euclid

```
function extended_gcd(a, b)
    x := 0    lastx := 1
    y := 1    lasty := 0
    while b ≠ 0
        quotient := a div b
        (a, b) := (b, a mod b)
        (x, lastx) := (lastx - quotient*x, x)
        (y, lasty) := (lasty - quotient*y, y)
    return (lastx, lasty)
```

# LCM

- Least Common Multiple (LCM) atau Kelipatan Persekutuan Terkecil (KPK) dari dua buah bilangan a dan b adalah bilangan terkecil m sedemikian hingga a | m dan b | m.
- Dihubungkan dengan GCD melalui identitas

$$\gcd(a,b) * lcm(a,b) = ab$$

- Implementasi?

# Multiple GCD & LCM

- Dapat dibuktikan bahwa
  gcd(a,b,c) = gcd(a, gcd(b,c)) = gcd(gcd(a,b),c)
- Jadi

$$\gcd(a_1, a_2, ..., a_n) = \gcd(a_1, \gcd(a_2, ..., a_n)), \text{ untuk } n > 2$$

- Serupa untuk LCM
- Implementasi ?

# Problems

1. Diberikan dua buah takaran air ukuran a liter dan b liter. Tentukan apakah kita dapat mengukur tepat c buah liter air dengan menggunakan a dan b, dan jika ya, minimal berapa kali penakaran?
2. Diberikan *n* buah takaran, $a_1$, ... ,$a_n$ tentukan apakah kita dapat mengukur tepat b liter, dan jika ya, minimal berapa kali pengukuran?

# Faktorisasi prima

- **Teorema Fundamental Aritmetika:** Setiap bilangan bulat positif n>1 dapat dinyatakan secara unik sebagai

$$n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$$

- dimana $p_i$ adalah bilangan prima ke-i, $p_k$ adalah bilangan prima terbesar yang membagi n dan $a_i \geq 0$
- Contoh: $1000 = 2^3 3^0 5^3$ dan $3528 = 2^3 3^2 5^0 7^2$

# Faktorisasi Prima

- Dengan faktorisasi prima, maka kita dapat menyatakan berbagai sifat dalam teori bilangan dengan lebih intuitif
- Representasi prima dari sebuah bilangan *n* dapat dianggap sebagai sebuah *multiset* dengan bilangan-bilangan prima sebagai elemen-elemennya dan pangkat sebagai *multiplicity*-nya.

# Faktorisasi Prima

- Jika $n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$ dan $\mathrm{m} = p_1^{b_1} p_2^{b_2} \ldots p_k^{b_k}$
- Maka
  - $n \mid m \Leftrightarrow a_i \le b_i, 1 \le i \le k$
  - $\gcd(m,n) = p_1^{\min(a_1,b_1)} p_2^{\min(a_2,b_2)} \ldots p_k^{\min(a_k,b_k)}$
  - $lcm(m,n) = p_1^{\max(a_1,b_1)} p_2^{\max(a_2,b_2)} \ldots p_k^{\max(a_k,b_k)}$
- Contoh: GCD dan LCM dari $2^{10}3^{20}7^{15}$ dan $2^{15}3^{10}5^{20}$ berturut-turut adalah $2^{10}3^{10}$ dan $2^{15}3^{20}5^{20}7^{15}$

# Problems

1. Tentukan faktorisasi prima dari 20!
2. Ada berapa banyak 0 di belakang representasi desimal dari 200! ?
3. Buat sebuah algoritme untuk menghitung faktorisasi prima dari C(n, k) = n!/[k!(n-k)!]
4. Diketahui Q(n,k) = n!/k!. Diberikan bilangan-bilangan bulat $1 \le a \le b \le 10000$ dan $1 \le c \le d \le 10000$ dan sebuah bilangan bulat m, buat algoritme untuk menghitung
   gcd(Q(b,a), Q(d,c)) mod m

# Banyak pembagi, jumlah pembagi

- Misal
  - d(n) = banyak pembagi dari n, e.g., d(6) = 4
  - S(n) = jumlah pembagi dari n, e.g., s(6) = 12
- Jika $n = p_1^{a_1} p_2^{a_2} \ldots p_k^{a_k}$ maka

$$d(n) = (a_1 + 1)(a_2 + 1)\ldots(a_k + 1)$$

$$s(n) = \left( \frac{p_1^{a_1+1} - 1}{p_1 - 1} \right) \ldots \left( \frac{p_k^{a_k+1} - 1}{p_k - 1} \right)$$

- Contoh: Tentukan d(1344) dan s(1344)?

# Problems

1. **(OSP 2012)**: Bilangan bulat positif terkecil yang memiliki tepat 6 pembagi (termasuk 1 dan dirinya sendiri) adalah 12. Bilangan bulat positif terkecil yang memiliki tepat 30 pembagi adalah... (Generalize?)
2. Buat algoritme menghitung s(n) dan d(n)!
3. Diberikan bilangan bulat n, buat algoritme untuk menghitung d(n!) dan s(n!) [ Jika s(n!) terlalu besar, hitung nilainya mod sebuah bilangan m]

# Aritmetika modular

- Definisi:

$$a = b (\mathrm{mod}\, m) \Leftrightarrow m \mid a - b$$

*m* | *a* jika dan hanya jika *a = 0 (mod m)*

- Beberapa sifat dasar aritmetika modular:

Jika a = b (mod m), c =d (mod m), dan k adalah sembarang bilangan bulat, maka:

1. $a + k \equiv b + k \mod m$
2. $a + c \equiv b + d \mod m$
3. $ak \equiv bk \mod m$
4. $ac \equiv bd \mod m$
5. $a^k \equiv b^k \mod m$, *jika* $k > 0$

# Aritmetika modular

- Bagaimana dengan pembagian?
- Jika $ak = bk (\mathrm{mod}\, m)$

  maka

  $$\gcd(k, m) = 1 \rightarrow a = b (\mathrm{mod}\, m)$$
- Jadi, boleh membagi jika yang dibagi tidak punya faktor persekutuan dengan modulus

# Inverse Perkalian Modular

- Diketahui m, *a* dan *b*, maka *ax = b* (mod m) pasti memiliki solusi jika gcd(*a*,*m*) = 1.
- Untuk mencari *x*, gunakan *Extended Euclid* untuk mencari *s* dan *t* s.r.s.

  $$as + mt = \gcd(a, m) = 1$$
- Maka, jelas bahwa *as* = 1 *(mod* m) sehingga

  $$x = (as)x = s(ax) = sb (\mathrm{mod}\, m)$$

  adalah solusi yang diinginkan
- s disebut sebagai inverse perkalian dari *a* mod m

Contoh: Tentukan *x* bulat positif terkecil yang memenuhi 64x = 7 (mod 125)

# Pemangkatan modular

- Pemangkatan modular adalah permasalahan yang sering muncul, e.g.: pada bidang keamanan informasi (kriptografi, digital signature etc.)
- Kita perlu menghitung $a^n$ *mod* m, dimana *n* bisa sangat besar
- Perlu sebuah cara untuk menghitung dengan singkat cepat!
- Untungnya: tidak perlu menghitung nilai $a^n$ secara eksplisit, karena hasil setiap perkalian dapat langsung direduksi mod m
- Algoritme naive: $\Theta(n)$

# Pemangkatan Cepat

- Fast Modular Exponentation
- Dasar: $a^{2^k} = (...(a^2)...)^2$ [sebanyak $k$ kali]
- Prinsipnya: dekomposisi pangkat *n* sebagai penjumlahan dari pangkat dari 2, hitung setiap pemangkatan secara terpisah, lalu kalikan hasilnya
- E.g. Karena $13 = 2^3 + 2^2 + 2^0$ maka
  $a^{13} = a^{2^3} a^{2^2} a^{2^0} = ((a^2 a)^2)^2 a$
- Kita hanya perlu 5 operasi perkalian, dibandingkan dengan 12 dengan cara naive
- Berapa kompleksitasnya?

# Pemangkatan Cepat

```
long long int exp(long long int a, long long int n, int m)
{
    long long int result = 1;
    a %= m;
    while(n>0)
    {
     if(n%2==0)
     {
      a = (a*a)%m;
      n /=2;
     }
     else
     {
      result = result*a; result %= m;
      n--;
     }
    }
    return result;
}
```

# Periodisitas Pemangkatan Modular

- Pangkat modular bersifat *periodik* karena hanya ada *m* buah nilai *mod* m
- E.g.:

| n | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $3^n$ mod 5 | 3 | 4 | 2 | 1 | 3 | 4 | 2 |

- Jika *n* terlalu besar, maka untuk menghitung pemangkatan, lebih efisien untuk mencari *periodisitas*-nya

# Pemangkatan Modular

- Jika $a^k = 1 (\mathrm{mod}\, m)$ dan $x = y (\mathrm{mod}\, k)$ maka
  $a^x = a^y (\mathrm{mod}\, m)$ (Bukti?)
- Prinsip ini dapat digunakan untuk menghitung $a^n$ (mod $m$) untuk $n$ yang besar dengan lebih cepat
- Contoh: $3^4 = 1$ mod 5, sehingga untuk menghitung $3^{2010}$ mod 5, kita hitung bahwa 2010 = 2 mod 4, sehingg $3^{2010} = 3^2 = 4$ mod 5

# Fermat's Little Theorem

- Jika $a$ adalah bilangan bulat dan $p$ adalah bilangan prima, maka $a^p = a (\mathrm{mod}\, m)$
- Jika gcd($a$,$p$) = 1, maka $a^{p-1} = 1 (\mathrm{mod}\, m)$
- Contoh:
  - 3 dan 7 adalah prima → gcd(3,7) = 1
  - $3^6 = 1$ mod 7 dan 1000=4 (mod 6), sehingga $3^{1000} = 3^4 = 4$ mod 7
- Tentukan:
  - $8^{111}$ mod 11 ?
  - $11^{100}$ mod 41 ?

## Fermat's Little Theorem

- FLT dapat digunakan untuk mempermudah perhitungan inverse bilangan modulo bilangan prima
- Karena $a^{p-1} = 1 (\mathrm{mod}\ p)$ , maka $x=a^{p-2}$ adalah solusi dari $ax = 1 (\mathrm{mod}\ p)$
- Dengan kata lain, jika gcd($a$, $p$) = 1, maka $a^{p-2}$ adalah inverse dari $a$ modulo $p$
- Berapa inverse dari 5 mod 23 dan 12 mod 31?

## Euler's

- Generalisasi dari Fermat
- Definisikan $\phi(n)$ (*Euler's totient function*) sebagai banyaknya bilangan bulat positif $\leq$ n yang saling prima dengan n, atau:

$$\phi(n) = \left|\{x \mid 1 \leq x < n \wedge \gcd(x, n) = 1\}\right|$$

- E.g. $\phi(2) = 1$, $\phi(6) = 2$, $\phi(10) = 4$, $\phi(p) = p-1$ jika p adalah prima
- Maka $\gcd(a, m) = 1 \rightarrow a^{\phi(m)} = 1 (\mathrm{mod}\ m)$

# Euler

- Bagaimana cara menghitung $\phi$(n) ?
- Cara naive: periksa semua *x* = 1 s/d n, apakah gcd(x,n) = 1 [tidak efisien]
- Lebih baik: gunakan formula berikut

$$\phi(n) = n(1-\frac{1}{p_1})...(1-\frac{1}{p_k})$$

dimana $p_i$, $1 \leq i \leq k$ adalah faktor-faktor prima dari *n*

- Implementasikan!

# Problems

1. Hitung:
   a. $3^{1001}$ mod 8
   b. Dua digit terakhir dari $3^{125}$
   c. $4444^{4444}$ mod 9
2. Diketahui definisi fungsi f(a,n), a, n $\geq$ 1 sbb.

$$f(a,n) = \begin{cases} a, & \text{jika } n = 1 \\ a^{f(a,n-1)} & \text{selainnya} \end{cases}$$

   Tentukan nilai dari
   a. *f*(3, 5) mod 7 ?
   b. 2 digit terakhir dari *f(313, 313)* ?

# Representasi Posisional

- Untuk setiap bilangan bulat $b > 1$, kita dapat menulis setiap bilangan bulat $n \geq 1$ sebagai

$$n = a_{k-1} \times b^{k-1} + a_{k-2} \times b^{k-2} + \ldots + a_0 \times b^0$$

$$n = (a_{k-1} a_{k-2} \ldots a_0)_b$$

- *b* adalah "basis" bilangan. E.g. dalam notasi sehari-hari, *b*=10, misal:

  $1325 = (1325)_{10} = 1x10^3+3x10^2+2x10^1+5x10^0$
- Setiap representasi dalam basis *b* memiliki *b* kemungkinan digit, yaitu: 0 s/d b-1.

# Representasi Posisional

- Untuk melakukan konversi antar basis, e.g. Dari basis b1 ke b2 :
  - Tentukan nilai bilangan input pada basis b1
  - Gunakan metode div-mod untuk mengubah nilai bilangan ke basis b2
- Hanya mungkin jika nilai bilangan masih tertampung dalam tipe data asli (32/64 bit)
- Konversikan: $(1a3f)_{16}$ ke basis 6 !
- Implementasikan!

# Problems

1. Buktikan bahwa jumlah semua digit sebuah bilangan n dalam basis *b* = n mod (b-1), e.g. dalam basis 10, 1325 = 1+3+2+5 (mod 9)!
2. Tentukan sifat serupa untuk mod (b+1) !
3. Diberikan bilangan *n* dalam basis *b* dengan panjang hingga 1000 digit, dan sebuah bilangan bulat *m*. Buat algoritme untuk menghitung *n%m* dan menampilkan nilainya dalam basis *b* juga!
4. (**GCJ 2009 – 1C: All Your Base**)

# Chinese Remainder Theorem

- Diberikan sistem persamaan modular sebagai berikut:

$$x = 2 (\mathrm{mod}\, 3)$$
$$x = 3 (\mathrm{mod}\, 5)$$
$$x = 2 (\mathrm{mod}\, 7)$$

- Kita harus menentukan *x* terkecil yang memenuhi persamaan tersebut
- Bentuk seperti ini dapat diselesaikan dengan menggunakan Chinese Remainder Theorem (CRT)

# Chinese Remainder Theorem

- Bentuk Umum:

$$x = a_1 \pmod{m_1}$$
$$x = a_2 \pmod{m_2}$$
$$\vdots$$
$$x = a_n \pmod{m_n}$$

- dimana $m_i$ $1 \leq i \leq n$ adalah bilangan yang saling prima $(\gcd(m_i, m_j) = 1, \text{ jika } i \neq j)$

# Chinese Remainder Theorem

- Metode penyelesaian:
  1. Set $M = m_1 m_2 \ldots m_n$ dan $M_i = M / m_i$
  2. Hitung $b_i$ yaitu inverse dari $M_i$ modulo $m_i$ (dijamin ada karena gcd($m_i$, $M_i$) = 1, hitung dengan Extended Euclid/FLT)
  3. Solusi yang diinginkan adalah

$$x = a_1 M_1 b_1 + a_2 M_2 b_2 + \ldots + a_n M_n b_n \pmod{M}$$

# Problem

1. Tentukan solusi persamaan pada contoh berikut!

$$x = 2 (\mathrm{mod}\, 3)$$
$$x = 3 (\mathrm{mod}\, 5)$$
$$x = 2 (\mathrm{mod}\, 7)$$

2. Diberikan 3 buah bilangan prima: 5, 7 dan 3, perhatikan bahwa 5|55, 7|56 dan 3|57. Diberikan *r* buah bilangan prima berbeda, $p_i$ tentukan sebuah bilangan bulat positif terkecil n, s.r.s. n, n+1, ..., n+(r-1) berturut-turut habis dibagi oleh $p_i$, $1 \leq i \leq r$.

   Batas: $1 \leq r \leq 10$, $2 \leq p_i < 100$