**MODUL PERKULIAHAN**

**EDISI 1**

**MATEMATIKA DISKRIT**



Penulis :

Nelly Indriani Widiastuti S.Si., M.T.

JURUSAN TEKNIK INFORMATIKA

UNIVERSITAS KOMPUTER INDONESIA

BANDUNG

2011

|  |
| --- |
|  TEORI BILANGAN**7** |
| JUMLAH PERTEMUAN : 1 PERTEMUANTUJUAN INSTRUKSIONAL KHUSUS : |

**Materi :**

## **Pendahuluan**

Bilangan bulat adalah bilangan yang tidak mempunyai pecahan desimal, misalnya 8, 21, 8765, -34, 0. Berlawanan dengan bilangan bulat adalah bilangan riil yang mempunyai titik desimal, seperti 8.0, 34.25, 0.02.

**Sifat Pembagian pada Bilangan Bulat**

Misalkan *a* dan *b* bilangan bulat, *a* ≠ 0.

*a* **habis** **membagi** *b* (*a divides b*) jika terdapat bilangan bulat *c* sedemikian sehingga *b* = *ac*.

Notasi: *a* | *b* jika *b* = *ac*, *c* ∈ **Z** dan *a* ≠ 0.

**Contoh 1**:

4 | 12 karena 124 = 3 (bilangan bulat) atau 12 = 4 × 3. Tetapi 4 | 13 karena 134 = 3.25 (bukan bilangan bulat).

## **PEMBAGI BERSAMA TERBESAR**

Misalkan *a* dan *b* bilangan bulat tidak nol.

Pembagi bersama terbesar (PBB – **greatest common divisor** atau *gcd*) dari *a* dan *b* adalah bilangan bulat terbesar *d* sedemikian hingga *d* | *a* dan *d* | *b*.

Dalam hal ini kita nyatakan bahwa PBB(*a*, *b*) = *d*.

**Contoh 2.**

Faktor pembagi 45: 1, 3, 5, 9, 15, 45;

Faktor pembagi 36: 1, 2, 3, 4, 9, 12, 18, 36;

Faktor pembagi bersama 45 dan 36: 1, 3, 9

**PBB(45, 36) = 9.**

**Teorema 2.** Misalkan *m* dan *n* bilangan bulat, dengan syarat *n* > 0 sedemikian sehingga

*m* = *nq* + *r* , 0 ≤ *r* < *n*

maka PBB(*m*, *n*) = PBB(*n*, *r*)

**Contoh 3**: *m* = 60, *n* = 18,

60 = 18 ⋅ 3 + 12

maka PBB(60, 18) = PBB(18, 12) = 6

## **TEOREMA dan ALGORITMA EUCLIDEAN**

**Teorema 3 (Teorema Euclidean).** Misalkan *m* dan *n* bilangan bulat, *n* > 0. Jika *m* dibagi dengan *n* maka terdapat bilangan bulat unik *q* (*quotient*) dan *r* (*remainder*), sedemikian sehingga

*m* = *nq* + *r* (1)

dengan 0 ≤ *r* < *n*.

**Contoh 4.**

1987/97 = 20, sisa 47:

 1987 = 97 ⋅ 20 + 47

 –22/3 = –8, sisa 2:

 –22 = 3(–8) + 2

tetapi –22 = 3(–7) – 1 salah, karena *r* = –1 (syarat 0 ≤ *r* < *n)*

## **Algoritma Euclidean**

Tujuan: algoritma untuk mencari PBB dari dua buah bilangan bulat.

Penemu: Euclides, seorang matematikawan Yunani yang menuliskan algoritmanya tersebut dalam buku, *Element*.

Misalkan *m* dan *n* adalah bilangan bulat tak negatif dengan *m* ≥ *n*. Misalkan *r*0 = *m* dan *r*1 = *n*. Lakukan secara berturut-turut pembagian untuk memperoleh

 *r*0 = *r*1*q*1 + *r*2 0 ≤ *r*2 ≤ *r*1,

 *r*1 = *r*2*q*2 + *r*3 0 ≤ *r*3 ≤ *r*2,



 *rn*– 2 = *rn*–1 *qn*–1 + *rn* 0 ≤ *rn*≤ *rn*–1,

 *rn*–1 = *rnqn* + 0

Menurut Teorema 2,

 PBB(*m*, *n*) = PBB(*r*0, *r*1) = PBB(*r*1, *r*2) = … =

 PBB(*rn*– 2, *rn*– 1) = PBB(*rn*– 1, *rn*) = PBB(*rn*, 0) = *rn*

Jadi, PBB dari *m* dan *n* adalah sisa terakhir yang tidak nol dari runtunan pembagian tersebut

Diberikan dua buah bilangan bulat tak-negatif *m* dan *n* (*m* ≥ *n*). Algoritma Euclidean berikut mencari pembagi bersama terbesar dari *m* dan *n*.

## **Algoritma Euclidean**

1. Jika *n* = 0 maka

 *m* adalah PBB(*m*, *n*);

stop.

tetapi jika *n* ≠ 0,

 lanjutkan ke langkah 2.

2. Bagilah *m* dengan *n* dan misalkan *r* adalah sisanya.

3. Ganti nilai *m* dengan nilai *n* dan nilai *n* dengan nilai *r*, lalu ulang kembali ke langkah 1.

**Contoh 5.**

*m* = 80, *n* = 12 dan dipenuhi syarat *m* ≥ *n*



Sisa pembagian terakhir sebelum 0 adalah 4, maka PBB(80, 12) = 4.

## **Kombinasi Lanjar**

PBB(*a*,*b*) dapat dinyatakan sebagai **kombinasi lanjar** (*linear combination*) *a* dan *b* dengan dengan koefisien-koefisennya.

**Contoh 6**:

PBB(80, 12) = 4 ,

4 = (-1) ⋅ 80 + 7 ⋅ 12.

**Teorema 3.** Misalkan *a* dan *b* bilangan bulat positif, maka terdapat bilangan bulat *m* dan *n* sedemikian sehingga PBB(*a*, *b*) = *ma* + *nb*.

**Contoh 7**:

Nyatakan PBB(21, 45) sebagai kombinasi lanjar dari 21 dan 45.

Solusi:

45 = 2 (21) + 3

21 = 7 (3) + 0

Sisa pembagian terakhir sebelum 0 adalah 3, maka **PBB(45, 21) = 3**

Substitusi dengan persamaan–persamaan di atas menghasilkan:

**3 = 45 – 2 (21)**

merupakan kombinasi lanjar dari 45 dan 21

**Contoh 8:**

Nyatakan PBB(312, 70) sebagai kombinasi lanjar 312 dan 70.

Solusi: Terapkan algoritma Euclidean untuk memperoleh PBB(312, 70):

312 = 4 ⋅ 70 + 32 (i)

70 = 2 ⋅ 32 + 6 (ii)

32 = 5 ⋅ 6 + 2 (iii)

6 = 3 ⋅ 2 + 0 (iv)

Sisa pembagian terakhir sebelum 0 adalah 2, maka **PBB(312, 70) = 2**

Susun pembagian nomor (iii) dan (ii) masing-masing menjadi

2 = 32 – 5 ⋅ 6 (iv)

6 = 70 – 2 ⋅ 32 (v)

Sulihkan (v) ke dalam (iv) menjadi

2 = 32 – 5⋅(70 – 2⋅32) = 1⋅32 – 5⋅70 + 10⋅32 = 11 ⋅ 32 – 5 ⋅ 70 (vi)

Susun pembagian nomor (i) menjadi

32 = 312 – 4 ⋅ 70 (vii)

Sulihkan (vii) ke dalam (vi) menjadi

2 = 11 ⋅ 32 – 5 ⋅ 70 = 11 ⋅ (312 – 4 ⋅ 70) – 5 ⋅ 70 = 11 . 312 – 49 ⋅ 70

Jadi, PBB(312, 70) = 2 = 11 ⋅ 312 – 49 ⋅ 70

## **Relatif Prima**

Dua buah bilangan bulat *a* dan *b* dikatakan *relatif prima* jika PBB(*a*, *b*) = 1.

**Contoh 9.**

20 dan 3 relatif prima sebab PBB(20, 3) = 1.

7 dan 11 relatif prima karena PBB(7, 11) = 1.

20 dan 5 tidak relatif prima sebab PBB(20, 5) = 5 ≠ 1.

Jika *a* dan *b* relatif prima, maka terdapat bilangan bulat *m* dan *n* sedemikian sehingga

*ma + nb = 1*

**Contoh 10.**

Bilangan 20 dan 3 adalah relatif prima karena PBB(20, 3) =1, atau dapat ditulis

2 . 20 + (–13) . 3 = 1 (*m* = 2, *n* = –13)

Tetapi 20 dan 5 tidak relatif prima karena PBB(20, 5) = 5 ≠ 1 sehingga 20 dan 5 tidak dapat dinyatakan dalam *m* . 20 + *n* . 5 = 1.

## **Aritmetika Modulo**

Misalkan *a* dan *m* bilangan bulat (*m* > 0). Operasi ***a* mod *m*** (dibaca “*a* modulo *m*”) memberikan sisa jika *a* dibagi dengan *m*.

Notasi: *a* mod *m* = *r* sedemikian sehingga

*a* = *mq* + *r*, dengan 0 ≤ *r* < *m*.

*m* disebut **modulus** atau **modulo**, dan hasil aritmetika modulo *m* terletak di dalam himpunan {0, 1, 2, …, *m* – 1}.

**Contoh 11.**

Beberapa hasil operasi dengan operator modulo:

(i) 23 mod 5 = 3 (23 = 5 ⋅ 4 + 3)

(ii) 27 mod 3 = 0 (27 = 3 ⋅ 9 + 0)

(iii) 6 mod 8 = 6 (6 = 8 ⋅ 0 + 6)

(iv) 0 mod 12 = 0 (0 = 12 ⋅ 0 + 0)

(v) – 41 mod 9 = 4 (–41 = 9 (–5) + 4)

(vi) – 39 mod 13 = 0 (–39 = 13(–3) + 0)

*Penjelasan untuk (v):* Karena *a* negatif, bagi |*a*| dengan *m* mendapatkan sisa *r*’. Maka *a* mod *m* = *m* – *r*’ bila *r*’ ≠ 0. Jadi |– 41| mod 9 = 5, sehingga –41 mod 9 = 9 – 5 = 4.

## **Kongruen**

* Misalnya 38 mod 5 = 3 dan 13 mod 5 = 3, maka dikatakan 38 ≡ 13 (mod 5) (baca: 38 kongruen dengan 13 dalam modulo 5).
* Misalkan *a* dan *b* bilangan bulat dan *m* adalah bilangan > 0, maka ***a* ≡ *b* (mod *m*)** jika *m* habis membagi *a* – *b*.

Jika *a* tidak kongruen dengan *b* dalam modulus *m*, maka ditulis *a* ≡*/* *b* (mod *m*) .

**Contoh 12.**

17 ≡ 2 (mod 3) *( 3 habis membagi 17 – 2 = 15)*

–7 ≡ 15 (mod 11) *(11 habis membagi –7 – 15 = –22)*

12 ≡/ 2 (mod 7) *(7 tidak habis membagi 12 – 2 = 10 )*

–7 ≡/ 15 (mod 3) *(3 tidak habis membagi –7 – 15 = –22)*

*a* ≡ *b* (mod *m*) dalam bentuk “sama dengan” dapat dituliskan sebagai

*a* = *b* + *km* (*k* adalah bilangan bulat)

**Contoh 13.**

17 ≡ 2 (mod 3) 🡺 17 = 2 + 5 ⋅ 3

–7 ≡ 15 (mod 11) 🡺 –7 = 15 + (–2)11

*a* mod *m* = *r* dapat juga ditulis sebagai

*a* ≡ *r* (mod *m*)

**Contoh 14.**

(i) 23 mod 5 = 3 🡺 23 ≡ 3 (mod 5)

(ii) 27 mod 3 = 0 🡺 27 ≡ 0 (mod 3)

(iii) 6 mod 8 = 6 🡺 6 ≡ 6 (mod 8)

(iv) 0 mod 12 = 0 🡺 0 ≡ 0 (mod 12)

(v) – 41 mod 9 = 4 🡺 –41 ≡ 4 (mod 9)

(vi) – 39 mod 13 = 0 🡺 – 39 ≡ 0 (mod 13)

**Teorema 4.** Misalkan *m* adalah bilangan bulat positif.

Jika *a* ≡ *b* (mod *m*) dan *c* adalah sembarang bilangan bulat maka

(*a* + *c*) ≡ (*b* + *c*) (mod *m*)

*ac* ≡ *bc* (mod *m*)

*ap* ≡ *bp* (mod *m*) , *p* bilangan bulat tak-negatif

Jika *a* ≡ *b* (mod *m*) dan *c* ≡ *d* (mod *m*), maka

(*a* + *c*) ≡ (*b* + *d*) (mod *m*)

*ac* ≡ *bd* (mod *m*)

**Contoh 15.**

Misalkan 17 ≡ 2 (mod 3) dan 10 ≡ 4 (mod 3), maka menurut Teorema 4,

17 + 5 = 2 + 5 (mod 3) ⇔ 22 = 7 (mod 3)

17 . 5 = 5 ⋅ 2 (mod 3) ⇔ 85 = 10 (mod 3)

17 + 10 = 2 + 4 (mod 3) ⇔ 27 = 6 (mod 3)

17 . 10 = 2 ⋅ 4 (mod 3) ⇔ 170 = 8 (mod 3)

Teorema 4 tidak memasukkan operasi pembagian pada aritmetika modulo karena jika kedua ruas dibagi dengan bilangan bulat, maka kekongruenan tidak selalu dipenuhi.

**Contoh 16**:

10 ≡ 4 (mod 3) dapat dibagi dengan 2

karena 10/2 = 5 dan 4/2 = 2, dan 5 ≡ 2 (mod 3)

14 ≡ 8 (mod 6) tidak dapat dibagi dengan 2, karena 14/2 = 7 dan 8/2 = 4, tetapi 7 ≡/ 4 (mod 6).

**Contoh 17:**

Jika *a* ≡ *b* (mod *m*) dan *c* ≡ d (mod *m*) adalah sembarang bilangan bulat maka buktikan bahwa

*ac* ≡ *bd* (mod *m*)

Solusi

*a* ≡ *b* (mod *m*) 🡪 *a = b + k1m*

*c* ≡ *d* (mod *m*) 🡪 *c = d + k2m*

maka

*ac =* (*b + k1m*)*(d + k2m)*

⇔ *ac = bd + bk2m + dk1m + k1k2m2*

⇔ *ac = bd + Km*  dengan *K = bk2 + dk1 + k1k2m*

***ac*** ≡ ***bd* (mod *m*)** (terbukti)

## **Balikan Modulo**

Di dalam aritmetika bilangan riil, inversi (*inverse*) dari perkalian adakah pembagian.

Contoh: Inversi 4 adalah 1/4, sebab 4 × 1/4 = 1.

Di dalam aritmetika modulo, masalah menghitung inversi modulo lebih sukar.

Jika *a* dan *m* relatif prima dan *m* > 1, maka balikan (*invers*) dari *a* modulo *m* ada.

Balikan dari *a* modulo *m* adalah bilangan bulat *x* sedemikian sehingga

*xa* ≡ 1 (mod *m*)

Dalam notasi lainnya, *a*–1(mod *m*) = *x*

Bukti: *a* dan *m* relatif prima, jadi PBB(*a*, *m*) = 1, dan terdapat bilangan bulat *x* dan *y* sedemikian sehingga *xa + ym = 1*

yang mengimplikasikan bahwa *xa + ym ≡ 1 (mod m)*

Karena *ym* ≡ 0 (mod *m*), maka *xa ≡ 1 (mod m)*

Kekongruenan yang terakhir ini berarti bahwa *x* adalah balikan dari *a* modulo *m*.

Pembuktian di atas juga menceritakan bahwa untuk mencari balikan dari *a* modulo *m*, kita harus membuat kombinasi lanjar dari *a* dan *m* sama dengan 1.

Koefisien *a* dari kombinasi lanjar tersebut merupakan balikan dari *a* modulo *m*.

**Contoh 18.**

Tentukan balikan dari 4 (mod 9), 17 (mod 7), dan 18 (mod 10).

Solusi:

1. Karena PBB(4, 9) = 1, maka balikan dari 4 (mod 9) ada. Dari algoritma Euclidean diperoleh bahwa 9 = 2 ⋅ 4 + 1

Susun persamaan di atas menjadi –2 ⋅ 4 + 1 ⋅ 9 = 1

Dari persamaan terakhir ini kita peroleh –2 adalah balikan dari 4 modulo 9.

Periksa bahwa  –2 ⋅ 4 ≡ 1 (mod 9)

Catatan: setiap bilangan yang kongruen dengan –2 (**mod** 9)

juga adalah inversi dari 4, misalnya 7, –11, 16, dan seterusnya, karena

7 ≡ –2 (**mod** 9) (9 habis membagi 7 – (–2) = 9)

–11 ≡ –2 (**mod** 9) (9 habis membagi –11 – (–2) = –9)

16 ≡ –2 (**mod** 9) (9 habis membagi 16 – (–2) = 18)

1. Karena PBB(17, 7) = 1, maka balikan dari 17 (mod 7) ada. Dari algoritma Euclidean diperoleh rangkaian pembagian berikut:

17 = 2 ⋅ 7 + 3 (i)

7 = 2 ⋅ 3 + 1 (ii)

3 = 3 ⋅ 1 + 0 (iii) (yang berarti: PBB(17, 7) = 1) )

Susun (ii) menjadi:

1 = 7 – 2 ⋅ 3 (iv)

Susun (i) menjadi

3 = 17 – 2 ⋅ 7 (v)

Sulihkan (v) ke dalam (iv):

1 = 7 – 2 ⋅ (17 – 2 ⋅ 7) = 1 ⋅ 7 – 2 ⋅ 17 + 4 ⋅ 7 = 5 ⋅ 7 – 2 ⋅ 17

atau

–2 ⋅ 17 + 5 ⋅ 7 = 1

Dari persamaan terakhir diperoleh –2 adalah balikan dari 17 (mod 7)

–2 ⋅ 17 ≡ 1 (mod 7) (7 habis membagi –2 ⋅ 17 – 1 = –35)

1. Karena PBB(18, 10) = 2 ≠ 1, maka balikan dari 18 (mod 10) tidak ada.

**Cara lain menghitung balikan**

Ditanya: balikan dari *a* (mod *m*)

Misalkan *x* adalah balikan dari *a* (mod *m*), maka

*ax* ≡ 1 (mod *m*) (definisi balikan modulo)

atau dalam notasi ‘sama dengan’:

*ax* = 1 + *km*

atau

*x =* (1 + *km*)/*a*

Cobakan untuk *k* = 0, 1, 2, … dan *k* = -1, -2, …

Solusinya adalah semua bilangan bulat yang memenuhi.

**Contoh 19**:

Balikan dari 4 (mod 9) adalah *x* sedemikian sehingga 4*x* ≡ 1 (mod 9)

4*x* ≡ 1 (mod 9) 🡪 4*x* = 1 + 9*k 🡪 x =* (1 + 9*k*)/4

Untuk *k* = 0 🡪 *x* tidak bulat

*k* = 1 🡪 *x* tidak bulat

*k* = 2 🡪 *x* tidak bulat

*k* = 3 🡪 *x* = (1 + 9 . 3)/4 = 7

*k* = -1 🡪 *x* = (1 + 9. –1)/4 = -2

Balikan dari 4 (mod 9) adalah 7 (mod 9), -2 (mod 9), dst

**Latihan**

Tentukan semua balikan dari 9 (mod 11).

Solusi:

Misalkan 9-1 (mod 11) = *x*

Maka 9*x* ≡ 1 (mod 11) atau 9*x* = 1 + 11*k* atau

*x* = (1 + 11*k*)/9

Dengan mencoba semua nilai *k* yang bulat (*k* = 0, -1, -2, ..., 1, 2, ...) maka

diperoleh *x* = 5. Semua bilangan lain yang kongruen dengan 5 (mod 11) juga merupakan solusi, yaitu –6, 16, 27, ...

## **Kongruen Lanjar**

Kekongruenan lanjar berbentuk:

*ax* ≡ *b* (mod *m*)

(*m* > 0, *a* dan *b* sembarang bilangan bulat, dan *x* adalah peubah bilangan bulat).

Pemecahan: *ax* = *b* + *km 🡺*

(Cobakan untuk *k* = 0, 1, 2, … dan *k* = –1, –2, … yang menghasilkan *x* sebagai bilangan bulat)

**Cara lain menghitung solusi
*ax* ≡ *b* (mod *m*)**

Seperti dalam persamaan biasa,

4*x* = 12 🡪 kalikan setiap ruas dengan 1/4 (yaitu invers 4),

maka 1/4 . 4*x* = 12 . 1/4

*x* = 3

4*x* ≡ 3 (mod 9) 🡪 kalikan setiap ruas dengan balikan dari 4 (mod 9) (dalam hal ini sudah kita hitung, yaitu –2)

(-2). 4*x* ≡ (-2).3 (mod 9)

-8*x* ≡ -6 (mod 9)

Karena –8 ≡ 1 (mod 9), maka *x* ≡ -6 (mod 9). Semua blangan bulat yang kongruen dengan –6 (mod 9) adalah solusinya, yitu 3, 12, …, dan –6, -15, …

**Latihan**

Sebuah bilangan bulat jika dibagi dengan 3 bersisa 2 dan jika ia dibagi dengan 5 bersisa 3. Berapakah bilangan bulat tersebut

Solusi

Misal : bilangan bulat = *x*

*x* mod 3 = 2 🡪 *x* ≡ 2 (mod 3)

*x* mod 5 = 3 🡪 *x* ≡ 3 (mod 5)

Jadi, terdapat sistem kekongruenan:

*x* ≡ 2 (mod 3) (i)

*x* ≡ 3 (mod 5) (ii)

Untuk kongruen pertama:

*x* = 2 + 3*k*1 (iii)

Substitusikan (iii) ke dalam (ii):

2 + 3*k*1 ≡ 3 (mod 5) 🡪 3*k*1 ≡ 1 (mod 5)

diperoleh

*k*1 ≡ 2 (mod 5) atau *k*1 = 2 + 5*k*2

*x* = 2 + 3*k*1

 = 2 + 3 (2 + 5*k*2)

 = 2 + 6 + 15*k*2

 = 8 + 15*k2* atau

*x* ≡ 8 (mod 15)

Semua nilai *x* yang kongruen dengan 8 (mod 15) adalah solusinya, yaitu *x* = 8,  *x* = 23,

*x*= 38, …,  *x* = -7, dst

## **Chinese Remainder Problem**

Pada abad pertama, seorang matematikawan China yang bernama Sun Tse mengajukan pertanyaan sebagai berikut:

*Tentukan sebuah bilangan bulat yang bila dibagi dengan 5 menyisakan 3, bila dibagi 7 menyisakan 5, dan bila dibagi 11 menyisakan 7.*

Misakan bilangan bulat tersebut = *x*. Formulasikan kedalam sistem kongruen lanjar:

*x* ≡ 3 (mod 5)

*x* ≡ 5 (mod 7)

*x* ≡ 7 (mod 11)

**Teorema 5. (*Chinese Remainder Theorem*)** Misalkan *m*1, *m*2, …, *mn* adalah bilangan bulat positif sedemikian sehingga PBB(*mi*, *mj*) = 1 untuk *i* ≠ *j*. Maka sistem kongruen lanjar

*x* ≡ *ak* (mod *mk*)

mempunyai sebuah solusi unik dalam modulo *m* = *m*1 ⋅ *m*2 ⋅ … ⋅ *mn*.

Solusi unik ini mudah dibuktikan sebagai berikut. Solusi tersebut dalam modulo:

*m* = *m*1 ⋅ *m*2 ⋅ *m*3 = 5 ⋅ 7 ⋅ 11 = 5 ⋅ 77 = 11 ⋅ 35.

Karena 77 . 3 ≡ 1 (mod 5),

55 ⋅ 6 ≡ 1 (mod 7),

35 ⋅ 6 ≡ 1 (mod 11),

maka solusi unik dari sistem kongruen tersebut adalah

*x* ≡ 3 ⋅ 77 ⋅ 3 + 5 ⋅ 55 ⋅ 6 + 7 ⋅ 35 ⋅ 6 (mod 385)

 ≡ 3813 (mod 385) ≡ 348 (mod 385)

## **BILANGAN PRIMA**

Bilangan bulat positif *p* (*p* > 1) disebut bilangan prima jika pembaginya hanya 1 dan *p*.

Contoh: 23

Karena bilangan prima harus lebih besar dari 1, maka barisan bilangan prima dimulai dari 2, yaitu 2, 3, 5, 7, 11, 13, ….

Seluruh bilangan prima adalah bilangan ganjil, kecuali 2 yang merupakan bilangan genap.

Bilangan selain prima disebut bilangan **komposit** (*composite*). Misalnya 20 adalah bilangan komposit karena 20 dapat dibagi oleh 2, 4, 5, dan 10, selain 1 dan 20 sendiri.

**Teorema 6. (*The Fundamental Theorem of Arithmetic*).** Setiap bilangan bulat positif yang lebih besar atau sama dengan 2 dapat dinyatakan sebagai perkalian satu atau lebih bilangan prima.

**Contoh 20.**

9 = 3 × 3

100 = 2 × 2 × 5 × 5

13 = 13 (atau 1 × 13)

Tes bilangan prima:

bagi *n* dengan sejumlah bilangan prima, mulai dari 2, 3, … , bilangan prima ≤ √*n*.

Jika *n* habis dibagi dengan salah satu dari bilangan prima tersebut, maka *n* adalah bilangan komposit, tetapi jika *n* tidak habis dibagi oleh semua bilangan prima tersebut, maka *n*

adalah bilangan prima.

**Contoh 21.**

Tes apakah (i) 171 dan (ii) 199 merupakan bilangan prima atau komposit.

Solusi *:*

√171 = 13.077. Bilangan prima yang ≤ √171 adalah 2, 3, 5, 7, 11, 13.

Karena 171 habis dibagi 3, maka 171 adalah bilangan komposit.

√199 = 14.107. Bilangan prima yang ≤ √199 adalah 2, 3, 5, 7, 11, 13.

Karena 199 tidak habis dibagi 2, 3, 5, 7, 11, dan 13, maka 199 adalah bilangan prima.

**Teorema 6** (**Teorema Fermat**). Jika *p* adalah bilangan prima dan *a* adalah bilangan bulat yang tidak habis dibagi dengan *p*, yaitu PBB(*a*, *p*) = 1, maka

*ap*–1 ≡ 1 (mod *p*)

**Contoh 22.**

Tes apakah 17 dan 21 bilangan prima atau bukan

Ambil *a* = 2 karena PBB(17, 2) = 1 dan PBB(21, 2) = 1.

217–1 = 65536 ≡ 1 (mod 17)

karena 17 habis membagi 65536 – 1 = 65535

Jadi, 17 prima.

221–1 =1048576 ≡\ 1 (mod 21)

karena 21 tidak habis membagi 1048576 – 1 = 1048575.

Jadi, 21 bukan prima

Kelemahan Teorema Fermat: terdapat bilangan komposit *n* sedemikian sehingga 2*n*–1 ≡ 1 (mod *n*). Bilangan bulat seperti itu disebut bilangan **prima semu** (*pseudoprimes*).

**Contoh 23:**

341 adalah komposit (karena 341 = 11 ⋅ 31) sekaligus bilangan prima semu, karena menurut teorema Fermat, 2340 ≡ 1 (mod 341)

Untunglah bilangan prima semu relatif jarang terdapat.

Untuk bilangan bulat yang lebih kecil dari 1010 terdapat 455.052.512 bilangan prima, tapi hanya 14.884 buah yang merupakan bilangan prima semu terhadap basis 2.

## **APLIKASI TEORI BILANGAN**

***ISBN***

Kode ISBN terdiri dari 10 karakter, biasanya dikelompokkan dengan spasi atau garis, misalnya 0–3015–4561–9.

ISBN terdiri atas empat bagian kode:

kode yang mengidentifikasikan bahasa,

1. kode penerbit,
2. kode unik untuk buku tersebut,
3. karakter uji (angka atau huruf X (=10)).

Karakter uji dipilih sedemikian sehingga

Contoh: ISBN 0–3015–4561–8

0 : kode kelompok negara berbahasa Inggris,

3015 : kode penerbit

4561 : kode unik buku yang diterbitkan

8 : karakter uji.

Karakter uji ini didapatkan sebagai berikut:

1 ⋅ 0 + 2 ⋅ 3 + 3 ⋅ 0 + 4 ⋅ 1 + 5 ⋅ 5 + 6 ⋅ 4 +

7 ⋅ 5 + 8 ⋅ 6 + 9 ⋅ 1 = 151

Jadi, karakter ujinya adalah 151 mod 11 = 8.

**Fungsi *Hash***

Tujuan: pengalamatan di memori

Bentuk: *h*(*k*) = *k* mod *m*

*m* : jumlah lokasi memori yang tersedia

k : kunci (*integer*)

*h*(*k*) :lokasi memori untuk *record* dengan kunci *k*

Kolisi (*collision*) terjadi jika fungsi *hash* menghasilkan nilai h yang sama untuk k yang berbeda. Jika terjadi kolisi, cek elemen berikutnya yang kosong.

Fungsi *hash* juga digunakan untuk me-*locate* elemen yang dicari.

**Kriptografi**

Dari Bahasa Yunani yang artinya “*secret writing*”

Definisi: kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan.

**Pesan**: data atau informasi yang dapat dibaca dan dimengerti maknanya.

Nama lain: **plainteks** (*plaintext*)

Pesan dapat berupa: teks, gambar, audio, video.

Pesan ada yang dikirim atau disimpan di dalam media penyimpanan.

**Algoritma kriptografi (*cipher*)**

**Cipherteks** (*ciphertext*): pesan yang telah disandikan sehingga tidak memiliki makna lagi.

Tujuan : agar pesan tidak dapat dimengerti maknanya oleh pihak lain.

Cipherteks harus dapat diubah kembali ke plainteks semula

**Enkripsi** (*encryption*): proses menyandikan plainteks menjadi ciphertek.

**Dekripsi** (*decryption*): Proses mengembalikan cipherteks menjadi plainteksnya.

Aturan untuk enkripsi dan dekripsi fungsi matematika yang digunakan untuk enkripsi dan dekripsi.

**Kunci**: parameter yang digunakan untuk transformasi *enciphering* dan *dechipering*

Kunci bersifat rahasia, sedangkan algoritma kriptografi tidak rahasia

Contoh:

Plainteks:

culik anak itu jam 11 siang

Cipherteks: t^$gfUi89rewoFpfdWqL:p[uTcxZ

**Aplikasi Kriptografi**

* Pengiriman data melalui saluran komunikasi (*data encryption on motion*).
* Penyimpanan data di dalam *disk storage* (*data encryption at rest*)

Data ditransmisikan dalam bentuk chiperteks. Di tempat penerima chiperteks dikembalikan lagi menjadi plainteks. Data di dalam media penyimpanan komputer (seperti *hard* *disk*) disimpan dalam bentuk chiperteks. Untuk membacanya, hanya orang yang berhak yang dapat mengembalikan chiperteks menjadi plainteks.

Contoh enkripsi pada dokumen dalam bentuk Notasi Matematis

Misalkan:

*C* = chiperteks

*P* = plainteks dilambangkan

Fungsi enkripsi *E* memetakan *P* ke *C*,

*E*(*P*) = *C*

Fungsi dekripsi *D* memetakan *C* ke *P*,

*D*(*C*) = *P*

Dengan menggunakan kunci *K*, maka fungsi enkripsi dan dekripsi menjadi

*EK*(*P*) = *C*

*DK*(*C*) = *P*

dan kedua fungsi ini memenuhi

*DK*(*EK*(*P*)) = *P*

* Jika kunci enkripsi sama dengan kunci dekripsi, maka sistem kriptografinya disebut **sistem** **simetri** atau **sistem** **konvensional**. Algoritma kriptografinya disebut algoritma simetri atau algoritma konvensional .

Contoh algoritma simetri:

*DES* (*Data Encyption Standard*)

* Jika kunci enkripsi tidak sama dengan kunci dekripsi, maka sistem kriptografinya disebut **sistem** **nirsimetri** (*asymmetric system*)

Nama lain: **sistem** **kriptografi kunci-publik**

karena, kunci enkripsi bersifat publik (*public key*) sedangkan kunci dekripsi bersifat rahasia (*private key*).

Pengirim pesan menggunakan kunci publik si penerima pesan untuk mengenkripsi pesan. Penerima pesan mendekripsi pesan dengan kunci privatnya sendiri.

Contoh algoritmai: *RSA*

***Caesar Cipher***

Tiap huruf alfabet digeser 3 huruf ke kanan

*pi*: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

*ci* : **D E F G H I J K L M N O P Q R S T U V W X Y Z A B C**

Contoh:

Plainteks: AWASI ASTERIX DAN TEMANNYA OBELIX

Cipherteks: **DZDVL DVWHULA GDQ WHPDQQBA REHOLA**

Misalkan *A* = 0, *B* = 1, …, *Z* = 25, maka secara matematis caesar *cipher* dirumuskan sebagai berikut:

Enkripsi: *ci* = *E*(*pi*) = (*pi* + 3) mod 26

Dekripsi: *pi* = *D*(*ci*) = (*ci* – 3) mod 26

*p*1 = ‘A’ = 0 🡪 *c*1 = *E*(0) = (0 + 3) mod 26 = 3 = ‘D’

*p*2 = ‘W’ = 22 🡪 *c*2 = *E*(22) = (22 + 3) mod 26 = 25 = ‘Z’

*p*3 = ‘A’ = 0 🡪 *c*3 = *E*(0) = (0 + 3) mod 26 = 3 = ‘D’

*p*4 = ‘S’ = 18 🡪 *c*4 = *E*(18) = (18 + 3) mod 26 = 21 = ‘V’ dst…

 Alternatif lain: gunakan tabel substitusi

Jika pergeseran huruf sejauh *k*, maka:

Enkripsi: *ci* = *E*(*pi*) = (*pi* + *k*) mod 26

Dekripsi: *pi* = *D*(*ci*) = (*ci* – *k*) mod 26

*k* = kunci rahasia

***Algoritma RSA***

Ditemukan oleh tiga peneliti dari *MIT* (*Massachussets Institute of Technology*), yaitu Ron Rivest, Adi Shamir, dan Len Adleman, pada tahun 1976. Termasuk algoritma kriptografi nirsimetri. Setiap pengguna mempunya sepasan kunci:

Kunci publik: untuk enkripsi

Kunci privat: untuk dekripsi

Kunci publik tidak rahasia (diktehui semua orang), kunci privat rahasia (hanya diketahui pemilik kunci saja)

**Pembangkitan pasangan kunci**

Pilih dua bilangan prima, *a* dan *b* (rahasia)

Hitung *n* = *a* *b*. Besaran *n* tidak perlu dirahasiakan.

Hitung *m* = (*a* – 1)(*b* – 1).

Pilih sebuah bilangan bulat untuk kunci publik, sebut

namanya *e*, yang relatif prima terhadap *m*.

Hitung kunci dekripsi, *d*, melalui *d* ≡ 1 (**mod** *m*).

**Enkripsi**

Nyatakan pesan menjadi blok-blok plainteks: *p*1, *p*2, *p*3, … (harus dipenuhi persyaratan bahwa nilai *pi*

harus terletak dalam himpunan nilai 0, 1, 2, …, *n* – 1

untuk menjamin hasil perhitungan tidak berada di luar himpunan)

Hitung blok cipherteks *ci* untuk blok plainteks *pi*

dengan persamaan

*ci* = *pie* **mod** *n*

yang dalam hal ini, *e* adalah kunci publik.

**Dekripsi**

Proses dekripsi dilakukan dengan menggunakan persamaan

*pi* = *cid* **mod** *n*,

yang dalam hal ini, *d* adalah kunci privat.

**Contoh 24:**

Misalkan *a* = 47 dan *b* = 71 (keduanya prima), maka dapat dihitung

*n* = *a* × *b* = 3337

*m* = (*a* – 1)×(*b* – 1) = 3220.

Pilih kunci publik *e* = 79 (yang relatif prima dengan 3220 karena pembagi bersama terbesarnya adalah 1). Nilai *e* dan *n* dapat dipublikasikan ke umum. Selanjutnya akan dihitung kunci dekripsi *d* dengan kekongruenan:

*e* × *d* ≡ 1 (mod *m*)

Dengan mencoba nilai-nilai *k* = 1, 2, 3, …, diperoleh nilai *d* yang bulat adalah 1019. Ini adalah kunci dekripsi.

Misalkan plainteks *P* = HARI INI

atau dalam desimal ASCII: 7265827332737873

Pecah *P* menjadi blok yang lebih kecil (misal 3 digit):

*p*1 = 726 *p*4 = 273

*p*2 = 582 *p*5 = 787

*p*3 = 733 *p*6 = 003

*Enkripsi setiap blok:*

*c*1 = 72679 mod 3337 = 215

*c*2 = 58279 mod 3337 = 776

dst untuk sisa blok lainnya

Keluaran: chiperteks *C* = 215 776 1743 933 1731 158.

*Dekripsi (menggunakan kunci privat d = 1019)*

*p*1 = 2151019 mod 3337 = 726

*p*2 =7761019 mod 3337 = 582

dst untuk sisi blok lainnya

Keluaran: plainteks *P* = 7265827332737873

yang dalam ASCII karakternya adalah HARI INI.

***Kekuatan dan Keamanan RSA***

* Kekuatan algoritma *RSA* terletak pada tingkat kesulitan dalam memfaktorkan bilangan non prima menjadi faktor primanya, yang dalam hal ini *n* = *a* × *b*.
* Sekali *n* berhasil difaktorkan menjadi *a* dan *b*, maka *m* = (*a* – 1)×(*b* – 1) dapat dihitung. Selanjutnya, karena kunci enkripsi *e* diumumkan (tidak rahasia), maka kunci dekripsi *d* dapat dihitung dari persamaan *e* × *d* ≡ 1 (mod *m*). Ini berarti proses dekripsi dapat dilakukan oleh orang yang tidak berhak.
* Penemu algoritma *RSA* menyarankan nilai *a* dan *b* panjangnya lebih dari 100 digit. Dengan demikian hasil kali *n* = *a* × *b* akan berukuran lebih dari 200 digit.
* Menurut Rivest dan kawan-kawan, uasaha untuk mencari faktor bilangan 200 digit membutuhkan waktu komputasi selama 4 milyar tahun! (dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik).