BAB 5

Algoritma dan Bilangan Bulat

Bulat air karena pembuluh, bulat kata karena mufakat.
(Pepatah adat Minangkabau)

Bab 5 ini akan membahas dua pokok bahasan: **algoritma** dan **bilangan bulat** (*integer*). Kedua pokok bahasan ini disatukan di dalam satu bab karena dalam pokok bahasan bilangan bulat kita akan mulai menggunakan terminologi algoritma untuk mendeskripsikan langkah-langkah penyelesaian masalah dengan suatu metode.

Algoritma menjadi penting karena ia merupakan jantung ilmu komputer. Banyak bahasan di dalam cabang-cabang ilmu komputer yang diacu dengan terminologi algoritma. Di dalam bab ini akan dijelaskan definisi algoritma, notasi yang digunakan, dan beberapa contoh algoritma dasar.

Pokok bahasan selanjutnya adalah bilangan bulat. Bilangan bulat atau *integer* memainkan peranan yang penting di dalam matematika diskrit. Sebagian besar (kalau tidak dapat dikatakan seluruhnya) pokok bahasan dalam matematika diskrit melibatkan bilangan bulat. Bahkan, cabang matematika yang bernama **teori bilangan** (*number theory*) mengkaji secara khusus bilangan bulat dan sifat-

sifatnya. Bilangan bulat sendiri merupakan objek diskrit, ini berlawanan dengan bilangan riil yang merupakan objek yang bersifat malar (*continue*).

5.1 Algoritma

Sebuah masalah dipecahkan dengan mendeksripsikan langkah-langkah penyelesaiannya. Misalnya masalah pengurutan (*sorting*) berikut: diberikan sejumlah bilangan bulat, tuliskan semua bilangan bulat tersebut dalam urutan yang menaik. Metode untuk pengurutan data diskrit sudah banyak ditemukan orang. Metode pengurutan sering digambarkan dalam sejumlah langkah terbatas yang mengarah pada solusi permasalahan. Urutan langkah-langkah penyelesaian masalah ini dinamakan algoritma.

DEFINISI 5.1 Algoritma adalah urutan logis langkah-langkah penyelesaian masalah yang disusun secara sistemastis.

Ditinjau dari asal usul kata, kata algoritma sendiri mempunyai sejarah yang aneh. Kata ini tidak muncul di dalam kamus Webster sampai akhir tahun 1957. Orang hanya menemukan kata *algorism* yang berarti proses menghitung dengan angka Arab [KNU73]. Anda dikatakan *algorist* jika anda menggunakan angka Arab. Para ahli bahasa berusaha menemukan asal kata *algorism* ini namun hasilnya kurang memuaskan. Akhirnya para ahli sejarah matematika menemukan asal mula kata tersebut. Kata *algorism* berasal dari nama penulis buku Arab yang terkenal, yaitu Abu Ja'far Muhammad ibnu Musa al-Khuwarizmi (al-Khuwarizmi dibaca orang Barat menjadi *algorism*). Al-Khuwarizmi menulis buku yang berjudul *Kitab al jabar wal-muqabala*, yang artinya "Buku pemugaran dan pengurangan" (*The book of restoration and reduction*). Dari judul buku itu kita juga memperoleh akar kata "aljabar" (*algebra*). Perubahan dari kata *algorism* menjadi *algorithm* muncul karena kata *algorism* sering dikelirukan dengan *arithmetic*, sehingga akhiran *-sm* berubah menjadi *-thm*. Karena perhitungan dengan angka Arab sudah menjadi hal yang sudah biasa/lumrah, maka lambat laun kata *algorithm* berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna aslinya [PAR95]. Dalam bahasa Indonesia, kata *algorithm* diserap menjadi algoritma.

Meskipun algoritma sering dikaitkan dengan ilmu komputer, namun sesungguhnya dalam kehidupan sehari-haripun banyak terdapat proses yang digambarkan dalam suatu algoritma. Cara-cara membuat kue atau masakan, misalnya dinyatakan dalam suatu resep. Misalnya resep membuat masakan Rendang Padang adalah contoh sebuah algoritma:

1. Potong daging sapi menjadi potongan-potongan dadu atau sesuai selera.
2. Haluskan bumbu berupa bawang merah, bawang putih, cabe merah, kunyit, laos, dan jahe.

3. Masukkan seluruh bumbu tadi ke dalam santan. Tambahkan dua buah daun jeruk, satu lembar daun kunyit, dan sebatang serai.
4. Masak santan di atas api sedang. Aduk terus hingga santan mendidih.
5. Masukkan daging sapi, dan kecilkan api. Sekali-sekali santan diaduk agar tidak pecah.
6. Jika sudah timbul minyak dan santan sudah kering, matikan api. Rendang siap dihidangkan.

Contoh-contoh algoritma yang lain dalam kehidupan sehari-hari misalnya pola pakaian, panduan praktikum, papan not balok, dan pengisian voucher ditunjukkan pada Tabel 5.1.

Tabel 5.1 Contoh-contoh Algoritma dalam Kehidupan Sehari-hari.

Proses	Algoritma	Contoh Langkah dalam Algoritma
1. Membuat kue	resep kue	Masukkan telur ke dalam wajan, kocok sampai mengembang
2. Membuat pakaian	pola pakaian	gunting kain dari pinggir kiri bawah ke arah kanan sejauh 5 cm
3. Praktikum reaksi kimia	panduan praktikum	campurkan 10 ml H_2SO4 dengan 15 ml NaOH
4. Merakit mobil	panduan merakit	sambungkan komponen A dengan komponen B
5. Kegiatan sehari-hari	jadwal harian	pukul 15.00 : tidur siang, pukul 16.00 : membuat PR
6. Memainkan musik	papan not balok	not balok
7. Mengisi *voucher* kartu prabayar telepon genggam (HP)	panduan pengisian	tekan nomor 888 masukkan nomor *voucher* 14 digit

5.2 Notasi Untuk Algoritma

Algoritma dapat dituliskan dalam berbagai notasi, misalnya dalam notasi kalimat-kalimat deksriptif seperti contoh resep masakan Rendang Padang di atas. Dengan notasi bergaya kalimat ini, deskripsi setiap langkah dijelaskan dengan bahasa sehari-hari secara gamblang. Setiap langkah biasanya diawali dengan kata kerja seperti 'baca', 'hitung', 'masukkan', 'bagi', 'ganti', dan sebagainya, sedangkan pernyataan bersyarat dinyatakan dengan 'jika ... maka ...'.

Sebagai contoh pertama, kita akan menuliskan algoritma untuk mencari elemen terbesar (*maximum*) dari sebuah himpunan yang beranggotakan n buah bilangan bulat. Bilangan-bilangan bulat tersebut dinyatakan sebagai $a_1, a_2, \ldots, a_n$. Elemen terbesar akan disimpan di dalam peubah (*variable*) yang bernama *maks*.

Algoritma Cari Elemen Terbesar

1. Asumsikan a_1 sebagai elemen terbesar sementara. Simpan a_1 ke dalam *maks*.
2. Bandingkan *maks* dengan elemen a_2. Jika a_2 lebih besar dari maks, maka nilai *maks* diganti dengan a_2.
3. Ulangi langkah 2 untuk elemen-elemen berikutnya (a_3, a_4, ..., a_n).
4. Berhenti jika tidak ada lagi elemen yang dibandingkan. Dalam hal ini, *maks* berisi nilai dari elemen terbesar.

Notasi algoritma dengan kalimat deskriptif bagus untuk algoritma yang pendek, namun untuk masalah yang algoritmanya besar, notasi ini tidak mangkus. Selain itu, notasi kalimat deskriptif kadang-kadang dianggap kurang bisa menjelaskan sebuah algoritma.

Selain dengan notasi deksriptif, algoritma juga dapat digambarkan dalam notasi bahasa komputer (lebih tepatnya bahasa pemrograman), misalnya dalam bahasa Pascal atau Bahasa C. Dalam Bahasa Pascal, algoritma mencari elemen terbesar ditulis seperti pada Algoritma 5.1.

```
procedure CariElemenTerbesar(a : array_integer; n : integer;
                             var maks : integer);
{ Mencari elemen terbesar di dalam array a[1..n]. Elemen terbesar akan
  disimpan di dalam maks. array_integer adalah tipe array yang sudah
  didefinisikan di dalam program utama dengan pendeklarasian berikut:
    const Nmaks = 1000;  { ukuran maksimum array }
    type array_integer = array[1..Nmaks] of integer ;
}
var
  i : integer;
begin
  maks := a[1];
  for i := 2 to n do
    if a[i] > maks then
      maks := a[i];
end;
```

Algoritma 5.1 Program Pascal untuk mencari elemen terbesar.

Sayangnya, setiap bahasa komputer memiliki aturan sintaks yang rumit yang membuat algoritma tersebut menjadi lebih sulit dipahami. Padahal, sebuah algoritma pada hakekatnya berisi abtraksi dari model penyelesaian masalah, sehingga algoritma seharusnya dibebaskan dari hal-hal teknis yang tidak perlu (misalnya tanda titik koma pada akhir setiap pernyataan, format masukan dan keluaran, dan lain-lain). Hal ini diperumit oleh kenyataan bahwa saat ini terdapat puluhan bahasa komputer, setiap bahasa tentu mempunyai aturan sintaks yang berbeda-beda.

Para ilmuwan komputer lebih menyukai menuliskan algoritma dalam notasi yang lebih praktis, yaitu notasi *pseudo-code*. *Pseudo-code* (*pseudo* artinya semu atau tidak sebenarnya) adalah notasi yang menyerupai notasi bahasa pemrograman tingkat tinggi, khususnya Bahasa *Pascal* dan *C*. Hasil pengamatan memperlihatkan bahwa bahasa pemrograman umumnya mempunyai notasi yang hampir mirip untuk beberapa instruksi, seperti notasi *if-then-else*, *while-do*, *repeat-until*, *read*, *write*, dan sebagainya. Berdasarkan pengamatan tersebut, maka beberapa penulis buku algoritma, termasuk penulis buku ini, mendefinisikan notasi algoritma yang disebut *pseudo-code* itu. Tidak seperti bahasa pemrograman yang direpotkan dengan tanda titik koma (*semicolon*), indeks, format keluaran, kata-kata khusus, dan sebagainya, sembarang versi *pseudo-code* dapat diterima asalkan perintahnya tidak membingungkan pembaca. Keuntungan menggunakan notasi *pseudo-code* adalah kemudahan mengkonversinya–lebih tepat disebut mentranslasi– ke notasi bahasa pemrograman, karena terdapat korespondensi antara setiap *pseudocode* dengan notasi bahasa pemrograman. Korespondensi ini dapat diwujudkan dengan tabel translasi dari notasi algoritmik ke notasi bahasa pemrograman apa pun.

Pseudo-code yang digunakan di dalam buku ini diadopsi dari Bahasa Pascal, namun tidak benar-benar mematuhi semua sintaks Pascal. Dengan menggunakan notasi *pseudo-code*, algoritma mencari elemen terbesar ditulis dengan notasi *pseudo-code* seperti ditunjukkan pada Algoritma 5.2.

```
procedure CariElemenTerbesar(input a1, a2, ..., an : integer,
                             output maks : integer)
{ Mencari elemen terbesar di antara elemen a1, a2, ..., an. Elemen
  terbesar akan disimpan di dalam maks.
  Masukan: a1, a2, ..., an
  Keluaran: maks
}
Deklarasi
  i : integer

Algoritma:
  maks ← a1
  for i ← 2 to n do
    if ai > maks then
      maks ← ai
    endif
  endfor
```

Algoritma 5.2 Algoritma mencari elemen terbesar dalam notasi *pseudo-code*.

Kata-kata yang digarisbawahi menyatakan kata kunci (*keywords*) yang nantinya berpadanan dengan kata kunci pada bahasa komputer yang dipilih untuk mentranslasikan algoritma tersebut. Kalimat yang diapit dengan pasangan kurung kurawal ({ dan }) menyatakan komentar. Komentar berguna untuk lebih memperjelas instruksi yang dituliskan. Peubah lokal yang digunakan di dalam algoritma dituliskan

pada bagian **Deklarasi**, sedangkan langkah-langkah penyelesian masalah dinyatakan di bagian **Algoritma**. Data masukan untuk algoritma dituliskan setelah kata input, sedangkan keluaran algoritma dituliskan sesudah kata output. Jika suatu peubah berfungsi sebagai masukan dan keluaran, maka peubah tersebut dituliskan seudah kata input/output. Karakter "←" menyatakan bahwa nilai di sebelah kanannya diiisikan ke dalam peubah di sebelah kirinya.

5.3 Beberapa Contoh Algoritma

Di bawah ini disajikan beberapa contoh algoritma. Algoritma pertama mempertukarkan nilai dari dua buah peubah. Algoritma kedua mencari elemen tertentu di antara sekumpulan nilai bilangan bulat. Sedangkan algoritma ketiga adalah algoritma untuk mengurutkan sekumpulan bilangan bulat.

Algoritma mempertukarkan nilai dari dua buah peubah.

Diberikan dua buah peubah, x dan y. Nilai x dan y dipertukarkan sehingga x akan berisi nilai y, sedangkan y akan berisi nilai x yang lama. Misalnya,

sebelum pertukaran, $x = 10, y = 8$,
setelah pertukaran, $x = 8$ dan $y = 10$.

Dalam operasi pertukaran ini, kita membutuhkan sebuah peubah bantu, *temp*, sebagai tempat menampung sementara nilai dari salah satu peubah (x atau y). Algoritma pertukaran nilai tersebut ditunjukkan oleh Algoritma 5.3.

```
procedure Tukar(input/output x, y : integer)
{ Mempertukarkan nilai x dan y
  Masukan: x dan y
  Keluaran: x dan y
}
Deklarasi
  temp : integer

Algoritma:
  temp ← x
  x ← y
  y ← temp
```

Algoritma 5.3 Algoritma mempertukarkan nilai dari dua buah peubah.

Algoritma mencari nilai tertentu di dalam himpunan elemen.

Diberikan n buah bilangan bulat yang dinyatakan sebagai $a_1, a_2, \ldots, a_n$. Carilah apakah x terdapat di dalam himpunan bilangan bulat tersebut. Jika x ditemukan,

maka lokasi (indeks) elemen yang bernilai x disimpan di dalam peubah *idx*. Jika x tidak terdapat di dalam himpunan tersebut, maka *idx* diisi dengan nilai 0.

Algoritma pencarian yang paling sederhana adalah algoritma **pencarian beruntun** (*sequential search* atau *linear search*). Setiap elemen di dalam himpunan dibandingkan dengan x, mulai dari elemen pertama, a_1, sampai elemen yang bernilai sama dengan x ditemukan atau sampai semua elemen sudah habis diperiksa. Jika $a_k = x$, maka k adalah lokasi tempat x berada (*idx* diisi dengan k). Jika x tidak ditemukan (semua elemen sudah habis diperiksa), maka 0 diisikan ke dalam *idx*. Algoritma pencarian yang termaksud ditunjukkan oleh Algoritma 5.4.

```
procedure PencarianBeruntun(input a1, a2, ..., an : integer, x : integer,
                            output idx : integer)
{ Mencari x di dalam elemen a1, a2, ..., an. Lokasi (indeks elemen) tempat
  x ditemukan diisi ke dalam idx. Jika x tidak ditemukan, maka idx diisi
  dengan 0.
  Masukan: a1, a2, ..., an
  Keluaran: idx
}
Deklarasi
  i : integer

Algoritma:
  i ← 1
  while (i < n) and (ai ≠ x) do
     i ← i + 1
  endwhile
  { i = n or ai = x }

  if ai = x then    { x ditemukan }
     idx ← i
  else
     idx ← 0        { x tidak ditemukan }
  endif
```

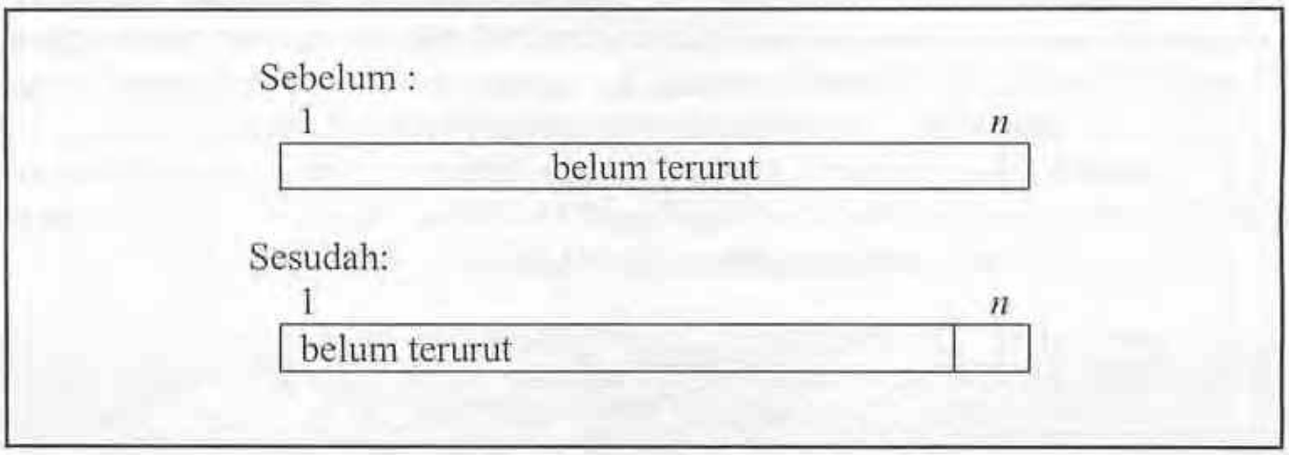
Algoritma 5.4 Algoritma pencarian beruntun.

Algoritma pengurutan

Diberikan n buah bilangan bulat yang dinyatakan sebagai a_1, a_2, ..., a_n. Urutkan semua bilangan bulat tersebut sedemikian sehingga $a_1 \le a_2 \le \ldots \le a_n$.

Pengurutan adalah persoalan yang kaya dengan solusi algoritma, karena saat ini terdapat puluhan algoritma pengurutan. Salah satu algoritma pengurutan adalah algoritma **pengurutan seleksi** (*selection sort*) Gagasan dasarnya adalah mencari elemen terbesar (atau terkecil), lalu menempatkan elemen terbesar (atau terkecil) itu pada awal atau akhir susunan (Gambar 5.1). Dua aktivitas ini disebut satu kali *pass*. Selanjutnya, elemen terujung tersebut "diisolasi" dan tidak disertakan pada proses selanjutnya. Untuk *pass* yang lainnya (seluruhnya ada $n - 1$ kali *pass*),

aktivitas yang sama diulang untuk elemen lain yang tersisa, yaitu memilih elemen terbesar (atau terkecil) berikutnya dan mempertukarkannya dengan elemen terujung dari bagian sisa.



Gambar 5.1 Bagian elemen yang terurut dan belum terurut pada metode Pengurutan Seleksi.

Algoritma pengurutan seleksi dituliskan di dalam Algoritma 5.5. Algoritam tersebut menghasilkan susunan yang menaik (*ascending order*). Perhatikan bahwa Algoritma 5.5 menggunakan Algoritma 5.2 (mencari elemen terbesar, tetapi yang dicatat hanya indeks elemen terbesar) dan Algoritma 5.3 (pertukaran dua buah nilai).

```
procedure PengurutanPilih(input n : integer,
                          input/output a1, a2, ..., an : integer,
                          )
{ Mengurutkan a1, a2, ..., an sehingga tersusun menaik dengan metode
  pengurutan maksimum.
  Masukan: a1, a2, ..., an
  Keluaran: a1, a2, ..., an (terurut menaik)
}
Deklarasi
    i     : integer    { pencacah jumlah pengulangan/pass }
    j     : integer    { pencacah untuk mencari nilai maksimum }
    imaks : integer    { indeks yang berisi nilai maksimum sementara }
    temp  : integer    { peubah bantu untuk pertukaran }

Algoritma:
    for i ← n downto 2 do   { jumlah pass sebanyak n - 1 }

        { cari elemen terbesar di antara a1, a2, ..., ai }
        imaks ← 1           { elemen pertama diasumsikan sebagai elemen
                              terbesar sementara }

        for j ← 2 to i do
          if aj > aimaks then
             imaks ← j
          endif
        endfor
```

```
    {pertukarkan a_imaks dengan a_i}
    temp ← a_i
    a_i ← a_imaks
    a_imaks ← temp
  endfor
```

Algoritma 5.5 Algoritma pengurutan pilih

5.4 Bilangan Bulat

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

Pada upabab-upabab berikut, kita akan mempelajari aritmetika bilangan bulat yang berhubungan dengan sifat pembagian. Jika diberikan dua buah bilangan bulat a dan b, apakah "a habis membagi b"? (kebalikan dari pertanyaan tersebut adalah: apakah "b kelipatan a"?

Sifat pembagian pada bilangan bulat melahirkan konsep-konsep seperti bilangan prima dan aritmetika modulo. Satu algoritma penting yang berhubungan dengan sifat pembagian ini adalah algoritma Euclidean. Baik bilangan prima, aritmetika modulo, dan algoritma Euclidean memainkan peranan yang penting dalam bidang kriptografi, yaitu ilmu yang mempelajari kerahasiaan pesan.

5.5 Sifat Pembagian pada Bilangan Bulat

Pembahasan bilangan bulat kita mulai dari sifat pembagian (*division*). Mengapa pembagian? Karena salah satu konsep bilangan bulat yang berguna dalam aritmetika komputer adalah bilangan prima. Bilangan prima adalah bilangan yang hanya habis dibagi oleh 1 dan dirinya sendiri. Bahkan, sembarang bilangan bulat positif dapat dinyatakan sebagai hasil perkalian satu atau lebih bilangan prima.

DEFINISI 5.2. Misalkan a dan b adalah dua buah bilangan bulat dengan syarat $a \neq 0$. Kita menyatakan bahwa a **habis membagi** b (a *divides* b) jika terdapat bilangan bulat c sedemikian sehingga $b = ac$.

Notasi: $a \mid b$ jika $b = ac$, $c \in \mathbf{Z}$ dan $a \neq 0$.

(ingatlah bahwa $\mathbf{Z}$ = himpunan bilangan bulat)

Dengan kata lain, jika b dibagi dengan a, maka hasil pembagiannya berupa bilangan bulat. Kadang-kadang pernyataan "a habis membagi b" ditulis juga "b **kelipatan** a".

Sebagai contoh, $4 \mid 12$ karena $12 \div 4 = 3$ (bilangan bulat) atau $12 = 4 \times 3$. Tetapi 4 tidak habis membagi 13 karena $13 \div 4 = 3.25$ (bukan bilangan bulat).

Secara umum, jika hasil pembagian bilangan bulat dinyatakan sebagai bilangan bulat juga, maka sembarang bilangan bulat bila dibagi dengan suatu bilangan bulat positif, maka selalu terdapat (1) hasil bagi dan (2) sisa pembagian. Misalnya, $13 \div 4$ memberikan hasil bagi 3 dan sisa 1. Kasus khusus, jika a habis membagi b, maka sisa pembagian adalah 0, misalnya $12 \div 4$ memberikan hasil bagi 3 dan sisa 0. Perhatikan juga bahwa sisa hasil pembagian selalu lebih besar atau sama dengan nol tetapi lebih kecil dari pembagi. Sifat ini kita tuangkan dalam Teorema 5.1 berikut.

TEOREMA 5.1. Misalkan m dan n adalah dua buah bilangan bulat dengan syarat $n > 0$. Jika m dibagi dengan n maka terdapat dua buah bilangan bulat unik q (*quotient*) dan r (*remainder*), sedemikian sehingga

$$m = nq + r \qquad (5.1)$$

dengan $0 \le r < n$.

Teorema 5.1 sering disebut juga **teorema Euclidean** (dari nama ilmuwan Yunani yang bernama Euclid, lahir pada tahun 350 sebelum Masehi). Bilangan n disebut **pembagi** (*divisor*), m disebut **yang dibagi** (*dividend*), q disebut **hasil bagi** (*quotient*), dan r disebut **sisa** (*remainder*). Notasi yang digunakan untuk mengekspresikan hasil bagi dan sisa adalah dengan menggunakan operator *mod* dan *div* seperti berikut:

$$q = m \textbf{ div } n, \qquad r = m \textbf{ mod } r$$

Contoh 5.1

1987 jika dibagi dengan 97 memberikan hasil bagi 20 dan sisa 47. Jadi, kita dapat menuliskan bahwa

$$1987 = 97 \,.\, 20 + 47$$

atau diekspresikan sebagai 1987 **div** 97 = 20 dan 1987 **mod** 97 = 47.
Begitu juga, jika –22 dibagi dengan 3 dapat dituliskan sebagai

$$-22 = 3(-8) + 2$$

atau diekspresikan sebagai –22 **div** 3 = –8 dan –22 **mod** 3 = 2.

Ingatlah bahwa sisa pembagian tidak boleh negatif, jadi kita tidak dapat menuliskan

$$-22 = 3(-7) - 1$$

karena $r = -1$ tidak memenuhi syarat $0 \le r < n$.
Sebaliknya, jika 24 dibagi dengan 3, maka kita dapat menuliskan

$$24 = 3 \cdot 8 + 0$$

karena $r = 0$ memenuhi syarat $0 \le r < n$. ■

5.6 Pembagi Bersama Terbesar

Dua buah bilangan bulat dapat memiliki faktor pembagi yang sama. Faktor pembagi bersama yang terpenting adalah faktor **pembagi bersama terbesar** (*greatest common divisor – gcd*)[1] atau PBB. Misalnya 45 memiliki faktor pembagi 1, 3, 5, 9, 15, dan 45 sendiri; sedangkan 36 memiliki faktor pembagi 1, 2, 3, 4, 9, 12, 18, dan 36 sendiri. Faktor pembagi bersama dari 45 dan 36 adalah 1, 3, 9, yang terbesar adalah 9 sehingga disimpulkan PBB(45, 36) = 9. Definisi formal dari PBB dinyatakan pada Definisi 5.3 berikut ini.

DEFINISI 5.3. Misalkan a dan b adalah dua buah bilangan bulat tidak nol. Pembagi bersama terbesar (PBB) dari a dan b adalah bilangan bulat terbesar d sedemikian sehingga $d \mid a$ dan $d \mid b$. Dalam hal ini kita nyatakan bahwa PBB$(a, b) = d$.

Sifat-sifat dari pembagi bersama terbesar dinyatakan dengan Teorema 5.2 [JOH97]:

TEOREMA 5.2. Misalkan a, b, dan c adalah bilangan bulat.
(a) Jika c adalah PBB dari a dan b, maka $c \mid (a + b)$
(b) Jika c adalah PBB dari a dan b, maka $c \mid (a - b)$
(c) Jika $c \mid a$, maka $c \mid ab$

Pembuktian untuk Teorema 5.2 hanya dikerjakan untuk yang (a) saja, selebihnya diserahkan kepada pembaca sebagai latihan.

Bukti:
(a) Karena c adalah PBB dari a dan b, maka $c \mid a$ dan $c \mid b$. Karena $c \mid a$, maka berarti

$$a = cd_1 \qquad (5.2)$$

untuk suatu bilangan bulat d_1. Begitu juga karena $c \mid b$, maka berarti

$$b = cd_2 \qquad (5.3)$$

[1] Di sekolah dasar dan menengah istilah "pembagi bersama terbesar" sering dinamakan "faktor persekutuan terbesar" atau FPB

untuk suatu bilangan bulat d_2. Jumlah dari (5.2) dan (5.3) adalah

$$a + b = cd_1 + cd_2 = c(d_1 + d_2) \qquad (5.4)$$

Dari (5.4) terlihat bahwa c habis membagi $a + b$. ■

Contoh 5.2

PBB dari 45 dan 36 adalah 9. Menurut Teorema 5.2,
(a) 9 habis membagi 45 + 36 = 81, atau 9 | (45 + 36)
(b) 9 habis membagi 45 – 36 = 9, atau 9 | (45 – 36)
(c) 9 habis membagi 45 · 36 = 1620, atau 9 | (45 . 36) ■

Dari Teorema 5.1 kita ingin menunjukkan bahwa PBB dari dua buah bilangan bulat sama dengan PBB dari salah satu bilangan bulat tersebut dengan sisa hasil pembagiannya. Hal ini dinyatakan pada Teorema 5.3 berikut ini.

TEOREMA 5.3. Misalkan m dan n adalah dua buah bilangan bulat dengan syarat $n > 0$ sedemikian sehingga

$$m = nq + r \quad , 0 \le r < n$$

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

Bukti untuk Teorema 5.3 tidak diberikan di sini dan ditinggalkan sebagai latihan buat pembaca.

Contoh 5.3

Jika 80 dibagi dengan 12 memberikan hasil 6 dan sisa 8, atau 80 = 12 . 6 + 8. Menurut Teorema 5.3,

PBB(80, 12) = PBB(12, 8) = 4

Jika 12 dibagi dengan 8 memberikan hasil 1 dan sisa 4, atau 12 = 8 . 1 + 4. Menurut Teorema 5.3,

PBB(12, 8) = PBB(8, 4) = 4

Jika 8 dibagi dengan 4 memberikan hasil 2 dan sisa 0, atau 8 = 4 . 2 + 0. Menurut Teorema 5.3,

PBB(8, 4) = PBB(4, 0) = 4

Dari runtunan perhitungan di atas, kita memperoleh bahwa

PBB(80, 12) = PBB(12, 8) = PBB(8, 4) = PBB(4, 0) = 4 ■

5.7 Algoritma Euclidean

Di dalam upabab 5.6 telah diperlihatkan bahwa untuk mencari PBB dari dua buah bilangan bulat m dan n, mula-mula kita mendaftarkan semua pembagi dari masing-masing m dan n, lalu memilih pembagi persekutuan yang bernilai terbesar. Di dalam upabab 5.7 ini diberikan metode yang mangkus untuk menemukan PBB, yang dikenal dengan nama **algoritma Euclidean**. Algoritma Euclidean sudah dikenal sejak berabad-abad yang lampau. Euclid, penemu algoritma Euclidean, adalah seorang matematikawan Yunani yang menuliskan algoritmanya tersebut dalam bukunya yang terkenal, *Element*.

Algoritma Euclidean didasarkan pada aplikasi Teorema 5.3 secara berturut-turut sampai kita menemukan sisa pembagian bernilai nol, dan contoh ilustrasi yang menarik untuk hal ini sudah diperlihatkan pada Contoh 5.3. Secara formal algoritma Euclidean kita dirumuskan sebagai berikut:

Misalkan m dan n adalah bilangan bulat tak negatif dengan $m \geq n$. Misalkan $r_0 = m$ dan $r_1 = n$. Lakukan secara berturut-turut pembagian (Teorema 5.1) untuk memperoleh

$$r_0 = r_1 q_1 + r_2 \qquad 0 \leq r_2 \leq r_1,$$

$$r_1 = r_2 q_2 + r_3 \qquad 0 \leq r_3 \leq r_2,$$

$$\vdots$$

$$r_{n-2} = r_{n-1}\, q_{n-1} + r_n \qquad 0 \leq r_n \leq r_{n-1},$$

$$r_{n-1} = r_n q_n + 0$$

Menurut Teorema 5.3,

$$\begin{aligned} \text{PBB}(m, n) &= \text{PBB}(r_0, r_1) = \text{PBB}(r_1, r_2) = \ldots = \text{PBB}(r_{n-2}, r_{n-1}) \\ &= \text{PBB}(r_{n-1}, r_n) = \text{PBB}(r_n, 0) = r_n \end{aligned}$$

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 ($m \geq n$). Algoritma Euclidean berikut mencari pembagi bersama terbesar, PBB, dari kedua bilangan tersebut, yaitu bilangan bulat positif terbesar yang habis membagi m dan n.

Algoritma Euclidean

1. Jika $n = 0$ maka
   m adalah PBB(m, n);
   stop.
   tetapi jika $n \neq 0$,
   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.

Catatan: jika $m \leq n$, maka pertukarkan terlebih dahulu nilai m dan n.

Contoh 5.4

Dengan mengambil contoh yang sama dengan Contoh 5.3, maka PBB dari 80 dan 12 dicari dengan algoritma Euclidean sebagai berikut:

$m = 80$, $n = 12$ dan dipenuhi syarat $m \geq n$

Karena $m = 12 \neq 0$, maka langkah instruksi 2 dikerjakan: 80 dibagi 12 memberikan hasil 6 dan sisa $r = 8$,

$$80 = 6 \cdot 12 + 8$$

Kerjakan langkah instruksi 3:

$m = 12$, $n = 8$

Kembali ke langkah instruksi 1, karena $n = 8 \neq 0$, maka langkah instruksi 2 dikerjakan: 12 dibagi 8 memberikan hasil 1 dan sisa $r = 4$,

$$12 = 1 \cdot 8 + 4$$

Kerjakan langkah instruksi 3:

$m = 8$, $n = 4$

Kembali ke langkah instruksi 1, karena $b = 4 \neq 0$, maka langkah instruksi 2 dikerjakan: 8 dibagi 4 memberikan hasil 2 dan sisa $r = 0$,

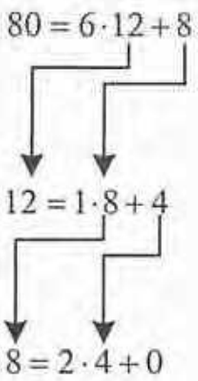
$$8 = 2 \cdot 4 + 0$$

Kerjakan langkah instruksi 3:

$m = 4$, $n = 0$

Kembali ke langkah instruksi 1, karena $b = 0$, maka PBB dari 80 dan 12 adalah nilai m terakhir, yaitu 4. Jadi PBB (80, 12) = 4.

Secara ringkas proses perhitungan dengan algoritma Euclidean di atas dinyatakan dalam runtunan pembagian berikut ini:



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

Dalam notasi *pseudo-code*, algoritma Euclidean kita tulis seperti pada Algoritma 5.6. Algoritma menerima masukan m dan n, dan menghasilkan keluaran PBB(m, n).

```
procedure Euclidean(input m, n : integer, output PBB : integer)
{ Mencari PBB(m, n) dengan syarat m dan n bilangan tak-negatif dan m ≥ n
  Masukan: m dan n dengan syarat m ≥ n dan m, n ≥ 0
  Keluaran: PBB(m, n)
}

Deklarasi
   r : integer

Algoritma:
   while n ≠ 0 do
      r ← m mod n
      m ← n
      n ← r
   endwhile
   { n = 0, maka PBB(m,n) = m }
   PBB ← m
```

Algoritma 5.6 Algoritma Euclidean

TEOREMA 5.4. Misalkan a dan b adalah dua buah bilangan bulat positif, maka terdapat bilangan bulat m dan n sedemikian sehingga PBB(a, b) = $ma + nb$.

Teorema 5.4 menyatakan bahwa PBB dua buah bilangan bulat a dan b dapat dinyatakan sebagai **kombinasi lanjar** (*linear combination*) dengan m dan n sebagai

koefisien-koefisennya. Misalnya PBB(80, 12) = 4, dan 4 = (-1) · 80 + 7 · 12. Di sini m = -1 dan n = 7.

Metode untuk menemukan kombinasi lanjar dari dua buah bilangan sama dengan PBB-nya adalah dengan melakukan pekerjaan pembagian secara mundur pada algoritma Euclidean. Perhatikan Contoh 5.5 berikut ini.

Contoh 5.5

Nyatakan PBB(312, 70) = 2 sebagai kombinasi lanjar dari 312 dan 70.

Penyelesaian:
Terapkan algoritma Euclidean untuk memperoleh PBB(312, 70) = 2:

$$312 = 4 \cdot 70 + 32 \quad \text{(i)}$$
$$70 = 2 \cdot 32 + 6 \quad \text{(ii)}$$
$$32 = 5 \cdot 6 + 2 \quad \text{(iii)}$$
$$6 = 3 \cdot 2 + 0 \quad \text{(iv)}$$

Susun pembagian nomor (iii) menjadi

$$2 = 32 - 5 \cdot 6 \quad \text{(iv)}$$

Susun pembagian nomor (ii) menjadi

$$6 = 70 - 2 \cdot 32 \quad \text{(v)}$$

Sulihkan (v) ke dalam (iv) menjadi

$$2 = 32 - 5 \cdot (70 - 2 \cdot 32) = 1 \cdot 32 - 5 \cdot 70 + 10 \cdot 32 = 11 \cdot 32 - 5 \cdot 70 \quad \text{(vi)}$$

Susun pembagian nomor (i) menjadi

$$32 = 312 - 4 \cdot 70 \quad \text{(vii)}$$

Sulihkan (vii) ke dalam (vi) menjadi

$$2 = 11 \cdot 32 - 5 \cdot 70 = 11 \cdot (312 - 4 \cdot 70) - 5 \cdot 70 = 11 \,.\, 312 - 49 \cdot 70$$

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

Relatif Prima

DEFINISI 5.4. Dua buah bilangan bulat a dan b dikatakan relatif prima (*relatively prime*) jika PBB(a, b) = 1.

Sebagai contoh, 20 dan 3 relatif prima sebab PBB(20, 3) = 1. Begitu juga 7 dan 11 relatif prima karena PBB(7, 11) = 1. Tetapi 20 dan 5 tidak relatif prima sebab PBB(20, 5) = 5 ≠ 1.

Jika a dan b relatif prima, maka menurut Teorema 5.4 kita dapat menemukan bilangan bulat m dan n sedemikian sehingga

$$ma + nb = 1 \qquad (5.5)$$

Contoh 5.6

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

$$2 \,.\, 20 + (-13) \,.\, 3 = 1$$

dengan m = 2 dan n = –13. Tetapi 20 dan 5 tidak relatif prima karena PBB(20, 5) = 5 ≠ 1 sehingga 20 dan 5 tidak dapat dinyatakan dalam $m \cdot 20 + n \cdot 5 = 1$. ■

5.8 Aritmetika Modulo

Aritmetika modulo *(modular arithmetic)* memainkan peranan yang penting dalam komputasi integer, khususnya pada aplikasi kriptografi. Operator yang digunakan pada aritmetika modulo adalah **mod**. Operator mod memberikan sisa pembagian. Misalnya 23 dibagi 5 memberikan hasil = 4 dan sisa = 3, sehingga kita tulis 23 mod 5 = 3 (operator mod sudah pernah kita gunakan pada Contoh 5.3). Definisi dari operator mod dinyatakan sebagai berikut:

DEFINISI 5.5. Misalkan a adalah bilangan bulat dan m adalah bilangan bulat > 0. Operasi a **mod** m (dibaca "a modulo m") memberikan sisa jika a dibagi dengan m. Dengan kata lain, a **mod** $m = r$ sedemikian sehingga $a = mq + r$, dengan $0 \le r < m$.

Notasi: a **mod** $m = r$ sedemikian sehingga $a = mq + r$, dengan $0 \le r < m$.

Bilangan m disebut **modulus** atau **modulo**, dan hasil aritmetika modulo m terletak di dalam himpunan $\{0, 1, 2, \ldots, m - 1\}$ (mengapa?).

Contoh 5.7

Beberapa hasil operasi dengan operator modulo:

(i) 23 **mod** 5 = 3 (karena 23 dibagi 5 memberikan hasil (q) = 4 dan sisa (r) = 3, atau ditulis sebagai 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 = 12 (0 = 12 · 0 + 12)
(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 $r' \neq 0$. Jadi $|-41|$ **mod** 9 = 5, sehingga –41 **mod** 9 = 9 – 5 = 4. ■

Jika a **mod** $m = 0$, maka dikatakan bahwa a adalah kelipatan dari m, yaitu a habis dibagi dengan m. Misalnya pada Contoh 5.7 di atas 27 **mod** 3 = 0, berarti 27 adalah kelipatan 3.

Kongruen

Kadang-kadang dua buah bilangan bulat, a dan b, mempunyai sisa yang sama jika dibagi dengan bilangan bulat positif m. Kita katakan bahwa a dan b **kongruen dalam modulo m**, dan dilambangkan sebagai

$$a \equiv b \ (\mathbf{mod}\ m)$$

(notasi '≡' dibaca 'kongruen')

Jika a tidak kongruen dengan b dalam modulus m, maka ditulis

$$a \not\equiv b \ (\mathbf{mod}\ m)$$

Misalnya 38 **mod** 5 = 3 dan 13 mod 5 = 3, maka 38 ≡ 13 (**mod** 5). Definisi formal dari kekongruenan dinyatakan sebagai berikut:

DEFINISI 5.6. Misalkan a dan b adalah bilangan bulat dan m adalah bilangan > 0, maka $a \equiv b$ (**mod** m) jika m habis membagi $a - b$.

Contoh 5.8

Bilangan 38 kongruen dengan 13 modulo 5 karena 5 membagi 38 – 13 = 25, sehingga dapat kita tulis bahwa 38 ≡ 13 (**mod** 5). Tetapi, 41 tidak kongruen dengan 30 modulo 5 karena 5 tidak habis membagi 41 – 30 = 11, sehingga dapat kita tulis 41 ≢ 30 (**mod** 5). Dengan cara yang sama, kita dapat menunjukkan bahwa

17 ≡ 2 (**mod** 3) (3 habis membagi 17 – 2 = 15 → 15 ÷ 3 = 5)
–7 ≡ 15 (**mod** 11) (11 habis membagi –7 – 15 = –22) → –22 ÷ 11 = 2)
12 ≢ 2 (**mod** 7) (7 tidak habis membagi 12 – 2 = 10)
–7 ≢ 15 (**mod** 3) (3 tidak habis membagi –7 – 15 = –22) ■

Kekongruenan $a \equiv b$ (**mod** m) dapat pula dituliskan dalam hubungan

$$a = b + km \tag{5.6}$$

yang dalam hal ini sembarang k adalah bilangan bulat. Pembuktiannya adalah sebagai berikut: menurut Definisi 5.6, $a \equiv b$ (**mod** m) jika $m \mid (a - b)$. Menurut Definisi 5.1, jika $m \mid (a - b)$, maka terdapat bilangan bulat k sedemikian sehingga $a - b = km$ atau $a = b + km$.

Contoh 5.9

Dari Contoh 5.6 di atas kita dapat menyatakan bahwa

$38 \equiv 13$ (**mod** 5) dapat ditulis sebagai $38 = 13 + 5 \cdot 5$
$17 \equiv 2$ (**mod** 3) dapat ditulis sebagai $17 = 2 + 5 \cdot 3$
$-7 \equiv 15$ (**mod** 11) dapat ditulis sebagai $-7 = 15 + (-2)11$ ■

Berdasarkan definisi aritmetika modulo (lihat Definisi 5.5), kita dapat menuliskan a mod $m = r$ sebagai

$$a \equiv r \text{ (mod } m)$$

Contoh 5.10

Beberapa hasil operasi dengan operator modulo berikut:
(i) 23 **mod** 5 = 3 dapat ditulis sebagai $23 \equiv 3$ (**mod** 5)
(ii) 27 **mod** 3 = 0 dapat ditulis sebagai $27 \equiv 0$ (**mod** 3)
(iii) 6 **mod** 8 = 6 dapat ditulis sebagai $6 \equiv 6$ (mod 8)
(iv) 0 **mod** 12 = 0 dapat ditulis sebagai $0 \equiv 0$ (**mod** 12)
(v) – 41 **mod** 9 = 4 dapat ditulis sebagai $-41 \equiv 4$ (**mod** 9)
(vi) – 39 **mod** 13 = 0 dapat ditulis sebagai $-39 \equiv 0$ (**mod** 13) ■

Sifat-sifat pengerjaan hitung pada aritmetika modulo, khususnya terhadap operasi perkalian dan penjumlahan, dinyatakan dalam Teorema 5.5 berikut:

TEOREMA 5.5. Misalkan m adalah bilangan bulat positif.

1. Jika $a \equiv b$ (**mod** m) dan c adalah sembarang bilangan bulat maka
   (i) $(a + c) \equiv (b + c)$ (**mod** m)
   (ii) $ac \equiv bc$ (**mod** m)
   (iii) $a^p \equiv b^p$ (**mod** m) untuk suatu bilanangan bulat tak negatif p.

2. Jika $a \equiv b$ (**mod** m) dan $c \equiv d$ (**mod** m), maka
   (i) $(a + c) \equiv (b + d)$ (mod m)
   (ii) $ac \equiv bd$ (**mod** m)

Bukti Teorema 5.5 hanya diperlihatkan untuk 1(ii) dan 2(i) saja. Bukti untuk 1(i), 1(iii), dan 2(ii) diserahkan sebagai latihan bagi pembaca.

Bukti:

1(ii) $a \equiv b$ (**mod** m) berarti:

$\Leftrightarrow a = b + km$ (dari persamaan 5.6)
$\Leftrightarrow a - b = km$
$\Leftrightarrow (a - b)c = ckm$ (kedua ruas dikalikan dengan c)
$\Leftrightarrow ac = bc + Km$ (dalam hal ini, $K = kc$)
$\Leftrightarrow ac \equiv bc$ (**mod** m) ■

2(i) $a \equiv b$ (**mod** m) $\Leftrightarrow a = b + k_1 m$
$c \equiv d$ (**mod** m) $\Leftrightarrow \underline{c = d + k_2 m}$ +
$\Leftrightarrow (a + c) = (b + d) + (k_1 + k_2)m$
$\Leftrightarrow (a + c) = (b + d) + km$ (dalam hal ini, $k = k_1 + k_2$)
$\Leftrightarrow (a + c) = (b + d)$ (**mod** m) ■

Contoh 5.11

Misalkan $17 \equiv 2$ (**mod** 3) dan $10 \equiv 4$ (**mod** 3), maka menurut Teorema 5.5,

$17 + 5 = 2 + 5$ (**mod** 3)	$\Leftrightarrow$	$22 = 7$ (**mod** 3)	(Teorema 5.5.1(i))
$17 \cdot 5 = 5 \cdot 2$ (**mod** 3)	$\Leftrightarrow$	$85 = 10$ (**mod** 3)	(Teorema 5.5.1(ii))
$17 + 10 = 2 + 4$ (**mod** 3)	$\Leftrightarrow$	$27 = 6$ (**mod** 3)	(Teorema 5.5.2(i))
$17 \cdot 10 = 2 \cdot 4$ (**mod** 3)	$\Leftrightarrow$	$170 = 8$ (**mod** 3)	(Teorema 5.5.2(ii)) ■

Perhatikanlah bahwa Teorema 5.5. tidak memasukkan operasi pembagian pada aritmetika modulo karena jika kedua ruas dibagi dengan bilangan bulat, maka kekongruenan tidak selalu dipenuhi. Misalnya:

(i) $10 \equiv 4$ (**mod** 3) dapat dibagi dengan 2 karena 10/2 = 5 dan 4/2 = 2, dan $5 \equiv 2$ (mod 3)

(ii) $14 \equiv 8$ (**mod** 6) tidak dapat dibagi dengan 2, karena 14/2 = 7 dan 8/2 = 4, tetapi $7 \not\equiv 4$ (**mod** 6).

Inversi Modulo (Modulo *Invers*)

Di dalam aritemetika bilangan riil, inversi (*inverse*) dari perkalian adakah pembagian. Misalnya inversi dari 4 adalah 1/4, karena $4 \times 1/4 = 1$. Di dalam aritmetika modulo, masalah menghitung inversi modulo lebih rumit.

Jika a dan m relatif prima dan $m > 1$, maka kita dapat menemukan inversi dari a modulo m. Inversi dari a modulo m adalah bilangan bulat $\bar{a}$ sedemikian sehingga

$$a\bar{a} \equiv 1 \ (\textbf{mod}\ m)$$

Pembuktian inversi modulo ini sangat mudah [ROS99]. Dari Definisi 5.4 tentang relatif prima diketahui bahwa PBB(a, m) = 1, dan menurut persamaan (5.5) terdapat bilangan bulat p dan q sedemikian sehingga

$$pa + qm = 1$$

yang mengimplikasikan bahwa

$$pa + qm \equiv 1 \ (\textbf{mod}\ m)$$

Karena $qm \equiv 0$ (**mod** m), maka

$$pa \equiv 1 \ (\textbf{mod}\ m)$$

Kekongruenan yang terakhir ini berarti bahwa p adalah inversi dari a modulo m. ■

Pembuktian di atas juga menceritakan bahwa untuk mencari inversi dari a modulo m, kita harus membuat kombinasi lanjar dari a dan m sama dengan 1. Koefisien a dari kombinasi lanjar tersebut merupakan inversi dari a modulo m. Perhatikan Contoh 5.12 berikut.

Contoh 5.12

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

Penyelesaian:

(a) Karena PBB(4, 9) = 1, maka inversi dari 4 (**mod** 9) ada. Dari algoritma Euclidean diperoleh bahwa

$$9 = 2 \cdot 4 + 1$$

Susun persamaan di atas menjadi

$$-2 \cdot 4 + 1 \cdot 9 = 1$$

Dari persamaan terakhir ini kita peroleh –2 adalah inversi dari 4 modulo 9. Periksalah bahwa

$$-2 \cdot 4 \equiv 1 \ (\textbf{mod}\ 9) \qquad (9 \text{ habis membagi } -2 \cdot 4 - 1 = -9)$$

Catatlah bahwa setiap bilangan yang kongruen dengan –2 modulo 9 juga adalah inversi dari 4, misalnya 7, –11, 16, dan seterusnya, karena

$7 \equiv -2$ (**mod** 9) (9 habis membagi $7 - (-2) = 9$)
$-11 \equiv -2$ (**mod** 9) (9 habis membagi $-11 - (-2) = -9$)
$16 \equiv -2$ (**mod** 9) (9 habis membagi $16 - (-2) = 18$)

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

$17 = 2 \cdot 7 + 3$ (i)
$7 = 2 \cdot 3 + 1$ (ii)
$3 = 3 \cdot 1 + 0$ (iii) (yang mengimplikasikan bahwa PBB(17, 7) = 1))

Susun (ii) menjadi:

$1 = 7 - 2 \cdot 3$ (iv)

Susun (i) menjadi

$3 = 17 - 2 \cdot 7$ (v)

Sulihkan (v) ke dalam (iv):

$$1 = 7 - 2 \cdot (17 - 2 \cdot 7) = 1 \cdot 7 - 2 \cdot 17 + 4 \cdot 7 = 5 \cdot 7 - 2 \cdot 17$$

atau

$$-2 \cdot 17 + 5 \cdot 7 = 1$$

Dari persamaan terakhir ini kita peroleh –2 adalah inversi dari 17 modulo 7.

$-2 \cdot 17 \equiv 1$ (**mod** 7) (7 habis membagi $-2 \cdot 17 - 1 = -35$)

(c) Karena PBB(18, 10) = $2 \neq 1$, maka inversi dari 18 (**mod** 10) tidak ada.

■

Metode lain yang cukup sederhana untuk menghitung inversi modulo adalah dengan melihat bahwa menurut persamaan (5.6) kekongruenan

$$a\bar{a} \equiv 1 \ (\textbf{mod}\ m)$$

dapat ditulis dalam hubungan

$$a\bar{a} = 1 + km$$

sehingga persoalan menemukan inversi modulo adalah ekivalen dengan menemukan $\bar{a}$ dan k sedemikian sehingga

$$\bar{a} = \frac{1+km}{a}$$

Sebagai ilustrasi, tinjau kembali Contoh 5.12 di atas, bahwa untuk inversi dari 4 (mod 9) adalah $\bar{a}$ sedemikian sehingga

$$4\bar{a} \equiv 1 \ (\textbf{mod } 9)$$

Inversi dari 4 (**mod** 9) adalah

$$\bar{a} = \frac{1+9k}{4}$$

Cobakan $k = 0, 1, 2, \ldots$ dan $k = -1, -2, \ldots$ ke dalam persamaan yang terakhir yang menghasilkan $\bar{a}$ sebagai bilangan bulat. Hasilnya adalah untuk $k = 3$ diperoleh $\bar{a} = 7$, untuk $k = -1$ diperoleh $\bar{a} = -2$, dan seterusnya.

Selain dengan kedua cara yang telah disebutkan di atas, ada metode lain yang banyak digunakan untuk mencari inversi modulo, yaitu **algoritma Euclidean yang diperluas** (*extended Euclidean algorithm*) [ROS03]

Kekongruenan Lanjar

Kekongruenan lanjar adalah kongruen yang berbentuk

$$ax \equiv b \ (\textbf{mod } m)$$

dengan m adalah bilangan bulat positif, a dan b sembarang bilangan bulat, dan x adalah peubah. Bentuk kongruen lanjar berarti menentukan nilai-nilai x yang memenuhi kekongruenan tersebut. Metode yang sederhana untuk mencari nilai-nilai x tersebut adalah dengan menggunakan persamaan (5.6). Menurut persamaan (5.6), $ax \equiv b$ (**mod** m) dapat ditulis dalam hubungan

$$ax = b + km$$

yang dapat disusun menjadi

$$x = \frac{b+km}{a}$$

dengan k adalah sembarang bilangan bulat. Cobakan nilai-nilai $k = 0, 1, 2, \ldots$ dan $k = -1, -2, \ldots$ ke dalam persamaan yang terakhir untuk menghasilkan x sebagai bilangan bulat.

Contoh 5.13

Tentukan solusi dari (i) $4x \equiv 3$ (**mod** 9) dan (ii) $2x \equiv 3$ (**mod** 4)

Penyelesaian:

(i) Kekongruenan $4x \equiv 3$ (**mod** 9) ekivalen dengan menemukan k dan x bilangan bulat sedemikian sehingga

$$x = \frac{3 + k \cdot 9}{4}$$

Nilai k bilangan bulat yang menghasilkan x bulat adalah untuk $k = 1$ diperoleh $x = 3$, untuk $k = 5$ diperoleh $x = 12$, untuk $k = -3$ diperoleh $x = -6$, untuk $k = -6$ diperoleh $x = -15$, dan seterusnya. Jadi, nilai-nilai x yang memenuhi $4x \equiv 3$ (**mod** 9) adalah 3, 12, ... dan $-6, -15, \ldots$

(ii) Kekongruenan $2x \equiv 3$ (**mod** 4) ekivalen dengan menemukan k dan x bilangan bulat sedemikian sehingga

$$x = \frac{3 + k \cdot 4}{2}$$

Karena $4k$ genap dan 3 ganjil maka penjumlahannya menghasilkan ganjil, sehingga hasil penjumlahan tersebut jika dibagi dengan 2 tidak menghasilkan bilangan bulat. Dengan kata lain, tidak ada nilai-nilai x yang memenuhi $2x \equiv 3$ (**mod** 4).

■

Metode lain untuk mencari solusi kekongruenan lanjar adalah dengan menggunakan inversi modulo. Caranya serupa dengan pencarian solusi pada persamaan lanjar biasa, seperti pada

$$4x = 12$$

Untuk mencari solusi persamaan di atas, kalikan kedua ruas dengan inversi perkalian dari 4, yaitu 1/4,

$$\begin{aligned} 1/4 \cdot 4x &= 1/4 \cdot 12 \\ 1 \cdot x &= 3 \\ x &= 3 \end{aligned}$$

Sekarang, terapkan metode seperti ini pada kekongruenan lanjar pada Contoh 5.13 di atas,

$$4x \equiv 3 \text{ (mod 9)}$$

Kalikan kedua ruas dengan inversi dari 4 (**mod** 9), yang dalam hal ini sudah dihitung pada Contoh 5.12, yaitu – 2:

$$-2 \cdot 4x \equiv -2 \cdot 3 \pmod 9$$
$$-8x \equiv -6 \pmod 9$$

Karena $-8 \equiv 1 \pmod 9$, maka $x \equiv -6 \pmod 9$. Jadi, solusi dari $4x \equiv 3 \pmod 9$ adalah bilangan bulat x sedemikian sehingga $x \equiv -6 \pmod 9$, yaitu 3, 12, ... dan $-6, -15, \ldots$
(Perhatikan bahwa

$3 \equiv -6 \pmod 9$, karena 9 habis membagi $3 - (-6) = 9$
$2 \equiv -6 \pmod 9$, karena 9 habis membagi $12 - (-6) = 18$
$-6 \equiv -6 \pmod 9$, karena 9 habis membagi $-6 - (-6) = 0$
$-15 \equiv -6 \pmod 9$, karena 9 habis membagi $-15 - (-6) = -9$

)

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.*

Pertanyaan Sun Tse dapat dirumuskan kedalam sistem kongruen lanjar:

$$x \equiv 3 \pmod 5$$
$$x \equiv 5 \pmod 7$$
$$x \equiv 7 \pmod{11}$$

Teorema *Chinese Remainder* berikut akan digunakan untuk menemukan solusi sistem kongruen lanjar seperti di atas.

TEOREMA 5.6. (*Chinese Remainder Theorem*) Misalkan $m_1, m_2, \ldots, m_n$ adalah bilangan bulat positif sedemikian sehingga PBB(m_i, m_j) = 1 untuk $i \neq j$. Maka sistem kongruen lanjar

$$x \equiv a_k \pmod{m_k}$$

mempunyai sebuah solusi unik dalam modulo $m = m_1 \cdot m_2 \cdot \ldots \cdot m_n$.

Contoh 5.14.

Tentukan solusi dari pertanyaan Sun Tse di atas.

Penyelesaian:
Menurut persamaan (5.6), kongruen pertama, $x \equiv 3 \pmod 5$, memberikan $x = 3 + 5k_1$ untuk beberapa nilai k. Sulihkan ini ke dalam kongruen kedua menjadi $3 + 5k_1 \equiv 5 \pmod$

7), dari sini kita peroleh $k_1 \equiv 6$ (**mod** 7), atau $k_1 = 6 + 7k_2$ untuk beberapa nilai k_2. Jadi kita mendapatkan $x = 3 + 5k_1 = 3 + 5(6 + 7k_2) = 33 + 35k_2$ yang mana memenuhi dua kongruen pertama. Jika x memenuhi kongruen yang ketiga, kita mempunyai $33 + 35k_2 \equiv 7$ (**mod** 11), yang mengakibatkan $k_2 \equiv 9$ (**mod** 11) atau $k_2 = 9 + 11k_3$. Sulihkan k_2 ini ke dalam kongruen yang ketiga menghasilkan $x = 33 + 35(9 + 11k_3) \equiv 348 + 385k_3$ (**mod** 11). Dengan demikian, $x \equiv 348$ (**mod** 385) yang memenuhi ketiga konruen tersebut. Dengan kata lain, 348 adalah solusi unik modulo 385. Catatlah bahwa $385 = 5 \cdot 7 \cdot 11$. Solusi unik ini mudah dibuktikan sebagai berikut. Solusi tersebut modulo $m = m_1 \cdot m_2 \cdot m_3 = 5 \cdot 7 \cdot 11 = 5 \cdot 77 = 11 \cdot 35$. Karena $77\ \ 3 \equiv 1$ (**mod** 5), $55 \cdot 6 \equiv 1$ (**mod** 7), dan $35 \cdot 6 \equiv 1$ (**mod** 11), solusi unik dari sistem kongruen tersebut adalah

$$x \equiv 3 \cdot 77 \cdot 3 + 5 \cdot 55 \cdot 6 + 7 \cdot 35 \cdot 6 \text{ (mod } 385)$$
$$\equiv 3813 \text{ (mod } 385) \equiv 348 \text{ (mod } 385)$$ ■

5.9 Bilangan Prima

Bilangan bulat positif yang mempunyai aplikasi penting dalam ilmu komputer dan matematika diskrit adalah bilangan prima. Bilangan prima adalah bilangan bulat positif yang lebih besar dari 1 yang hanya habis dibagi oleh 1 dan dirinya sendiri.

DEFINISI 5.7. Bilangan bulat positif p ($p > 1$) disebut bilangan prima jika pembaginya hanya 1 dan p.

Sebagai contoh, 23 adalah bilangan prima karena ia hanya habis dibagi oleh 1 dan 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 penting yang menyangkut bilangan prima dinyatakan oleh teorema yang terkenal dalam teori bilangan, yaitu **teorema fundamental aritmetik**, yang bunyinya adalah seperti di bawah ini.

TEOREMA 5.7 (*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.

Teorema 5.7 menyatakan bahwa baik bilangan prima maupun bilangan komposit, keduanya dapat dinyatakan sebagai perkalian dari satu atau lebih faktor prima. Misalnya,

$9 = 3 \times 3$ (2 buah faktor prima)
$100 = 2 \times 2 \times 5 \times 5$ (4 buah faktor prima)
$13 = 13$ (atau 1×13) (1 buah faktor prima)

Masalah lain yang juga penting adalah menguji apakah sebuah bilangan merupakan prima atau bukan. Teorema 5.6 menyatakan bahwa sembarang bilangan bulat positif habis dibagi oleh faktor-faktor primanya. Faktor prima dari sebuah bilangan selalu lebih kecil atau sama dengan akar kuadrat dari bilangan tersebut. Hal ini mudah ditunjukkan sebagai berikut: misalkan a adalah faktor prima dari n, dengan $1 < a < n$, maka a habis membagi n dengan hasil bagi b sedemikian sehingga $n = ab$. Nilai a dan b haruslah $\leq n$ agar

$$ab > \sqrt{n} \cdot \sqrt{n} = n$$

Dengan kata lain, faktor prima dari n selalu lebih kecil atau sama dengan $\sqrt{n}$. Hal ini dinyatakan dengan teorema berikut:

TEOREMA 5.8. Jika n adalah bilangan komposit, maka n mempunyai faktor prima yang lebih kecil atau sama dengan $\sqrt{n}$.

Untuk menguji apakah n merupakan bilangan prima atau komposit, kita cukup membagi n dengan sejumlah bilangan prima, mulai dari 2, 3, ... , bilangan prima $\leq \sqrt{n}$. 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 5.15

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

Penyelesaian:

(i) $\sqrt{171} = 13.077$. Bilangan prima yang $\leq \sqrt{171}$ adalah 2, 3, 5, 7, 11, 13. Karena 171 habis dibagi 3, maka 171 adalah bilangan komposit.

(ii) $\sqrt{199} = 14.107$. Bilangan prima yang $\leq \sqrt{199}$ adalah 2, 3, 5, 7, 11, 13. Karena 199 tidak habis dibagi 2, 3, 5, 7, 11, dan 13, maka 199 adalah bilangan prima.

■

Contoh 5.16

Temukan semua faktor prima dari 1617.

Penyelesaian:

Bagilah 1617 berturut-turut dengan barisan bilangan prima, mulai dari 2, 3, 5, 7, ...

2 tidak habis membagi 1617
3 habis membagi 1617, yaitu 1617/3 = 539

Selanjutnya, bagilah 539 dengan bilangan prima berturut-turut, dimulai dari 3, 5, ...

3 tidak habis membagi 539
5 tidak habis membagi 539
7 habis membagi 539, yaitu 539/7 = 77

Selanjutnya, bagilah 77 dengan bilangan prima berturut-turut, dimulai dari 7, 11, ...

7 habis membagi 77, yaitu 77/7 = 11

Karena 11 adalah bilangan prima, maka pencarian faktor prima dari 1617 dihentikan. Jadi, faktor prima dari 1617 adalah 3, 7, 7, dan 11, yaitu $1617 = 3 \times 7 \times 7 \times 11$. ■

Menguji keprimaan suatu bilangan bulat n dengan cara membaginya dengan sejumlah bilangan prima yang $\leq \sqrt{n}$ tentu tidak mangkus untuk n yang besar, karena kita harus menentukan terlebih dahulu semua bilangan prima yang lebih kecil dari $\sqrt{n}$. Untunglah terdapat metode lain yang dapat digunakan untuk menguji keprimaan suatu bilangan bulat, yang terkenal dengan **Teorema Fermat** (kadang-kadang dinamakan juga *Fermat's Little Theorem*). Fermat adalah seorang matematikawan Perancis pada tahun 1640.

TEOREMA 5.9 (Teorema Fermat). Jika p adalah bilangan prima dan a adalah bilangan bulat yang tidak habis dibagi dengan p, yaitu PBB$(a, p) = 1$, maka

$$a^{p-1} \equiv 1 \ (\mathbf{mod}\ p)$$

Contoh 5.17

Kita akan menguji apakah 17 dan 21 bilangan prima atau bukan. Di sini kita mengambil nilai $a = 2$ karena PBB(17, 2) = 1 dan PBB(21, 2) = 1. Untuk 17,

$$2^{17-1} = 65536 \equiv 1 \ (\mathbf{mod}\ 17)$$

karena 17 habis membagi 65536 – 1 = 65535 (yaitu, $65535 \div 17 = 3855$). Karena 17 memenuhi teorema Fermat, maka 17 adalah bilangan prima.
Untuk 21,

$$2^{21-1} = 1048576 \equiv\backslash\ 1 \ (\mathbf{mod}\ 21)$$

karena 21 tidak habis membagi 1048576 – 1 = 1048575. Karena 21 tidak memenuhi teorema Fermat, maka 21 bukan bilangan prima. ■

Sayangnya, teorema Fermat ini mengandung kekurangan sebab terdapat bilangan komposit n sedemikian sehingga $2^{n-1} \equiv 1$ (**mod** n). Bilangan bulat seperti itu disebut bilangan **prima semu** (*pseudoprimes*) terhadap basis 2. Misalnya

komposit 341 (yaitu 341 = 11 . 31) adalah bilangan prima semu terhadap basis 2 karena menurut teorema Fermat,

$$2^{340} \equiv 1 \ (\textbf{mod}\ 341)$$

Kita dapat menggunakan bilangan selain 2 sebagai basis menguji bilangan prima semu.

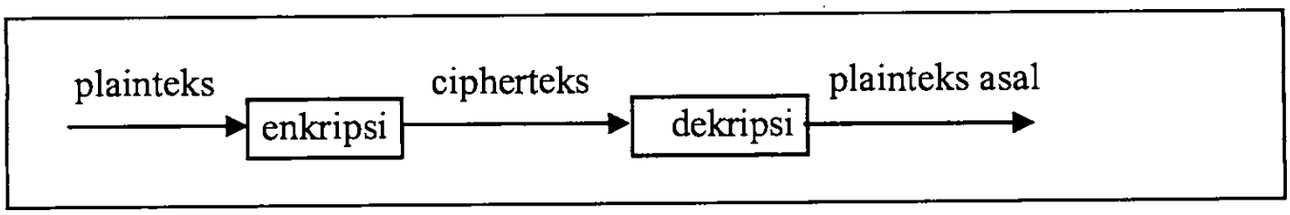
Untunglah bilangan prima semu relatif jarang terdapat [ROS99]. Untuk bilangan bulat yang lebih kecil dari 10^{10} terdapat 455.052.512 bilangan prima, tapi hanya 14.884 buah yang merupakan bilangan prima semu terhadap basis 2 [ROS03].

5.10 Kriptografi

Aritmetika modulo dan bilangan prima mempunyai banyak aplikasi dalam ilmu komputer, salah satu aplikasinya yang terpenting adalah **kriptografi**.

Kriptografi adalah ilmu sekaligus seni untuk menjaga keamanan pesan [SCH96]. Keamanan pesan diperoleh dengan menyandikannya menjadi pesan yang tidak mempunyai makna. Zaman sekarang ini kerahasiaan informasi menjadi sesuatu yang penting. Informasi yang rahasia perlu disembunyikan agar tidak diketahui oleh orang yang tidak berhak. Anda tentu tidak ingin nomor PIN kartu kredit atau kartu ATM anda diketahui orang, bukan? Atau, jika anda menulis sebuah pesan yang sifatnya rahasia, anda tidak ingin pesan tersebut dibaca oleh orang lain. Kriptografi dapat digunakan untuk menyamarkan informasi rahasia itu dari orang atau pihak yang tidak berhak membacanya.

Pesan yang dirahasiakan dinamakan **plainteks** (*plaintext*, artinya teks jelas yang dapat dimengerti), sedangkan pesan hasil penyandian disebut **cipherteks** (*ciphertext*, artinya teks tersandi). Pesan yang telah disandikan dapat dikembalikan lagi ke pesan aslinya hanya oleh orang yang berhak (orang yang berhak adalah orang yang mengetahui metode penyandian atau memiliki kunci penyandian). Proses menyandikan plainteks menjadi cipherteks disebut **enkripsi** (*encryption*) dan proses membalikkan cipherteks menjadi plainteksnya disebut **dekripsi** (*decryption*). Gambar 5.1 memperlihatkan diagram kedua proses yang dimaksud.



Gambar 5.1. Enkripsi dan Dekripsi

Sebagai contoh, sebuah pesan rahasia (plainteks) berikut:

```
uang disimpan di balik buku X
```

disandikan menjadi cipherteks dengan suatu teknik kriptografi tertentu menjadi:

```
j&kloP(d$gkhtpuBn%6^klp..t@
```

Cipherteks, meskipun tidak dirahasiakan, namun isinya sudah tidak jelas dan tidak dapat dimengerti maksudnya. Hanya orang yang berhak yang dapat mengeminversi pesan tidak jelas tersebut menjadi pesan semula.

Kriptografi digunakan untuk dua aplikasi, yaitu aplikasi pengiriman data melalui saluran komunikasi dan aplikasi penyimpanan data di dalam *disk storage*. Data ditransmisikan melalui saluran komunikasi dalam bentuk cipherteks. Di tempat penerima cipherteks dikembalikan lagi menjadi plainteks. Sedangkan untuk aplikasi penyimpanan, data di dalam media penyimpanan (seperti *hard disk*) disimpan dalam bentuk cipherteks. Untuk membacanya, hanya orang yang berhak yang dapat membalikkan cipherteks menjadi plainteks.

Contoh enkripsi dan dekripsi pada data tersimpan misalnya enkripsi pada dokumen plainteks yang bernama `plain.txt` yang isinya adalah:

```
Ketika saya berjalan-jalan di pantai,
saya menemukan banyak sekali kepiting
yang merangkak menuju laut. Mereka
adalah anak-anak kepiting yang baru
menetas dari dalam pasir. Naluri
mereka mengatakan bahwa laut adalah
tempat kehidupan mereka.
```

Misalkan hasil enkripsi `plain.txt` dengan metode kriptografi tertentu disimpan di dalam berkas `cipher.txt` yang isinya adalah sebagai berikut:

```
Ztâxzp/épêp/qtüyp{p}<yp{p}/sx/□p}âpx;
□□épêp/|t}t|äzp}/qp}êpz/étzp{x/zt□xâx
}v□□êp}v/|tüp}vzpz/|t}äyä/{päâ=/\tütz
p□□psp{pw/p}pz<p}pz/zt□xâx}v/êp}
v/qpüä□□|t}tâpé/spüx/sp{p|/□péxü=/]
p{äüx□□|ttüzp/|t}vpâpzp}/qpwâp/{päâ
/psp{pw□□ât|□pâ/ztwxsä□p}/|tützp=
```

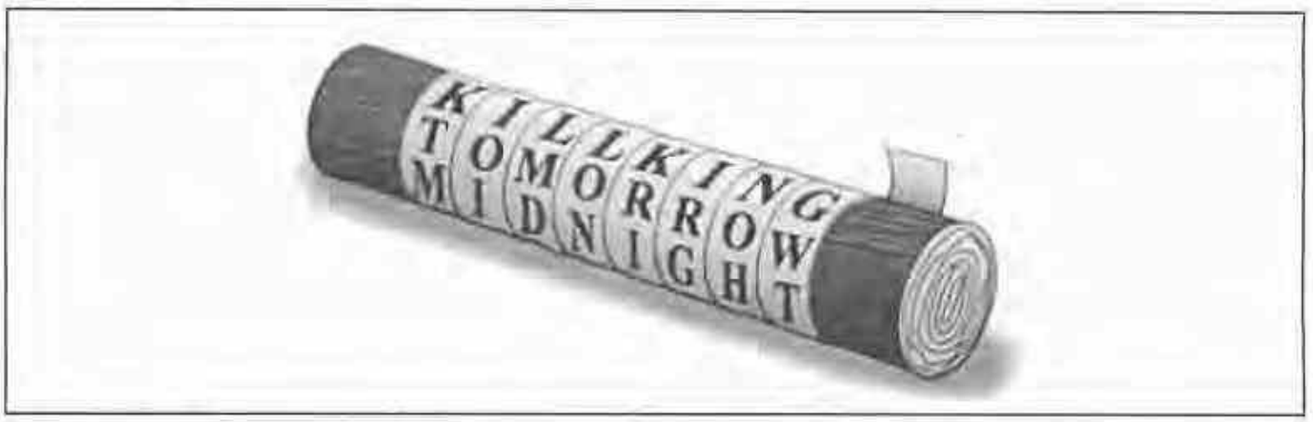
Hasil dekripsi terhadap berkas `cipher.txt` mengahsilkan berkas yang tepat sama dengan berkas `plain.txt` semula:

```
Ketika saya berjalan-jalan di pantai,
saya menemukan banyak sekali kepiting
yang merangkak menuju laut. Mereka
adalah anak-anak kepiting yang baru
menetas dari dalam pasir. Naluri
mereka mengatakan bahwa laut adalah
tempat kehidupan mereka.
```

Perdagangan elektronis (*e-commerce*) pun menggunakan kriptografi untuk menyandikan nomor PIN *customer*, karena pembayaran di dalam *e-commerce* umumnya menggunakan kartu kredit. Pengacakan (*scrambling*) siaran televisi melalui parabola, seperti pada siaran Piala Dunia 2002 di RCTI baru-baru ini, juga termasuk termasuk ke dalam aplikasi kriptografi. Di sini, penyandian dilakukan dengan mengacak susunan gambar.

Sejarah Kriptografi

Menurut sejarahnya, kriptografi sudah lama digunakan oleh tentara Sparta di Yunani pada permulaan tahun 400 SM. Mereka menggunakan alat yang disebut *scytale*. Alat ini terdiri dari sebuah pita panjang dari daun papyrus yang dililitkan pada sebatang silinder (Gambar 5.2). Pesan yang akan dikirim ditulis horizoantal (baris per baris). Bila pita dilepaskan, maka huruf-huruf di dalamnya telah tersusun membentuk pesan rahasia. Untuk membaca pesan, penerima melilitkan kembali silinder yang diameternya sama dengan diemeter silinder pengirim. Teknik kriptografi seperti ini dikenal dengan nama transposisi cipher, yang merupakan metode enkripsi tertua.



Gambar 5.2. *Scytale* yang digunakan oleh tentara Yunani untuk transposisi pesan

Kriptanalisis dan Kriptologi

Kriptanalisis (*cryptanalysis*) adalah ilmu dan seni untuk memecahkan cipherteks menjadi plainteks tanpa mengetahui *kunci* yang diberikan. Jadi, kriptanalisis adalah kebalikan dari kriptografi. Pelakunya disebut **kriptanalis**. **Kriptologi** (*cryptology*) adalah studi mengenai kriptografi dan kriptanalisis (*cryptanalyst*).

Jika seorang kriptografer menggunakan enkripsi untuk merahasiakan pesan dan mendekripsikannya kembali, maka seorang kriptanalis (*cryptanalyst*) mempelajari metode enkripsi dan cipherteks dengan tujuan menemukan plainteksnya. Baik kriptografer maupun kriptanalis menerjemahkan cipherteks menjadi plainteks, namun kriptografer bekerja atas legitimasi pengirim atau penerima pesan, sedangkan kriptanalis bekerja atas nama penyusup yang tidak berhak.

Notasi Matematis

Jika cipherteks dilambangkan dengan C dan plainteks dilambangkan dengan P, maka fungsi enkripsi E memetakan P ke C,

$$E(P) = C \tag{5.7}$$

Pada proses kebalikannya, fungsi dekripsi D memetakan C ke P,

$$D(C) = P \tag{5.8}$$

Dengan kata lain, D adalah fungsi inversi (*inverse*) dari E, atau $D = E^{-1}$. Seperti yang telah dijelaskan di dalam Bab 3, fungsi yang mempunyai inversi adalah fungsi yang berkoresponden satu-ke-satu. Sifat berkoreponden satu-ke-satu harus djamin pada fungsi enkripsi dan dekripsi, karena cipherteks harus dapat dikembalikan ke plainteks semula.

Karena proses enkripsi kemudian dekripsi mengeminversi pesan ke pesan asal, maka kesamaan berikut harus benar,

$$D(E(P)) = P \tag{5.9}$$

Algoritma kriptografi –atau *cipher*– adalah fungsi matematika yang digunakan untuk enkripsi dan dekripsi. Kekuatan suatu algoritma kriptografi diukur dari banyaknya kerja yang dibutuhkan untuk memecahkan data cipherteks menjadi plainteksnya. Kerja ini dapat diekivalenkan dengan waktu. Semakin banyak usaha yang diperlukan, yang berarti juga semakin lama waktu yang dibutuhkan, maka semakin kuat algoritma kriptografinya, yang berarti semakin aman digunakan untuk menyandikan pesan.

Jika kekuatan kriptografi ditentukan dengan menjaga kerahasiaan algoritmanya, maka algoritma kriptografinya dinamakan algoritma *restricted*. Misalkan di dalam sebuah kelompok orang mereka sepakat menyandikan setiap pesan pesan dengan algoritma yang sama. Algoritmanya adalah mempertukarkan pada setiap kata karakter pertama pdengan karakter kedua, karakter ketiga dengan karakter keempat dan seterusnya. Contohnya,

Plainteks: STRUKTUR DISKRIT
Cipherteks: TSURTKRU IDKSIRT

Untuk mendeksripsikan pesan, algoritma yang sama digunakan kembali. Sayangnya, algoritma *restricted* tidak cocok lagi saat ini. Bila salah seorang dari anggota keluar dari kelompok, maka algoritma penyandian pesan harus diubah lagi karena kerahasiaannya tidak dapat lagi diandalkan.

Kriptografi modern tidak lagi mendasarkan kekuatan pada algoritmanya. Jadi algoritmanya tidak dirahasiakan dan boleh diketahui oleh umum. Kekuatan kriptografinya terletak pada kunci, yaitu berupa deretan karakter atau bilangan bulat. Kunci ini dijaga kerahasiaannya, dan hanya orang yang mengetahui kunci dapat melakukan enkripsi dan dekripsi. Kunci di sini sama fungsinya dengan sandi-lewat (*password*) pada sistem komputer, PIN pada kartu ATM atau kartu kredit. Bedanya, jika sandi-lewat atau PIN bertujuan untuk otorisasi akses, maka kunci pada kriptografi digunakan pada proses enkripsi maupun dekripsi.

Misalnya pada *Caesar cipher*, yaitu teknik kriptografi yang digunakan oleh Kaisar Romawi, Julius Caesar, untuk menyandikan pesan yang ia kirim kepada para gubernurnya. Pada *Caesar cipher*, tiap huruf disubstitusi dengan huruf ketiga berikutnya dari susunan alfabet. Dalam hal ini kuncinya adalah jumlah pergeseran huruf (yaitu 3). Susunan alfabet setelah digeser sejauh 3 huruf adalah:

```
Plainteks:   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
Cipherteks:  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
```

Jadi, huruf A pada plainteks disubstitusi dengan D, huruf B disubstitusi dengan E, demikian seterusnya. Dengan mengkodekan setiap huruf alfabet dengan *integer*: '*A*'= 0, '*B*' = 1, ..., '*Z*' = 25, maka secara matematis pergeseran 3 huruf alfabet ekivalen dengan melakukan operasi modulo terhadap plainteks p menjadi cipherteks c dengan persamaan

$$c = E(p) = (p + 3) \bmod 26 \qquad (5.10)$$

Contoh 5.18

Pesan

```
AWASI ASTERIX DAN TEMANNYA OBELIX
```

disandikan dengan *Caesar Cipher*. Dengan mengkodekan 'A' = 0, 'B' = 1, ..., 'Z' = 25, maka cipherteks dihitung dengan persamaan 5.10 sebagai berikut:

p_1 = 'A' = 0 → $c_1 = E(0) = (0 + 3) \bmod 26 = 3$ = 'D'
p_2 = 'W' = 22 → $c_2 = E(22) = (22 + 3) \bmod 26 = 25$ = 'Z'
p_3 = 'A' = 0 → $c_3 = E(0) = (0 + 3) \bmod 26 = 3$ = 'D'
p_4 = 'S' = 18 → $c_4 = E(18) = (18 + 3) \bmod 26 = 21$ = 'V'
dst...

Bila keseluruhan perhitungan diselesaikan, maka diperoleh cipherteksnya adalah

```
DZDVL DVWHULA GDQ WHPDQQBA REHOLA
```

Alternatif lain, cipherteks juga dapat langsung diperoleh dengan menggunakan tabel pergeseran 3 huruf di atas, yaitu: A disubstitusidengan D, W disubstitusi dengan Z, A disubstitusi dengan D, W disubstitusi dengan V, dst. ■

Penerima pesan mengembalikan lagi cipherteks dengan operasi kebalikan yang secara matematis dapat dinyatakan dengan persamaan

$$p = D(c) = (c - 3) \bmod 26 \qquad (5.11)$$

Perhatikan bahwa D adalah inversi (*inverse*) dari fungsi E, yaitu $D(c) = E^{-1}(p)$. Sehingga, cipherteks

```
DZDVL DVWHULA GDQ WHPDQQBA REHOLA
```

dikembalikan menjadi plainteks asal dengan persamaan 5.11 menjadi

```
AWASI ASTERIX DAN TEMANNYA OBELIX
```

Secara umum, fungsi enkripsi dan dekripsi paad *Caesar Cipher* dapat dibuat lebih umum dengan menggeser huruf alfabet sejauh k sehingga

$$c = E(p) = (p + K) \bmod 26 \qquad (5.12)$$

untuk fungsi enkripsi dan

$$p = D(c) = (c - K) \bmod 26 \qquad (5.13)$$

untuk fungsi dekripsi. Pada kedua persamamaan terakhir ini, K berlaku sebagai kunci rahasia.

Secara matematis, pada sistem kriptografi yang menggunakan kunci K, maka fungsi enkripsi dan dekripsi menjadi

$$E_{K1}(P) = C \qquad (5.14)$$

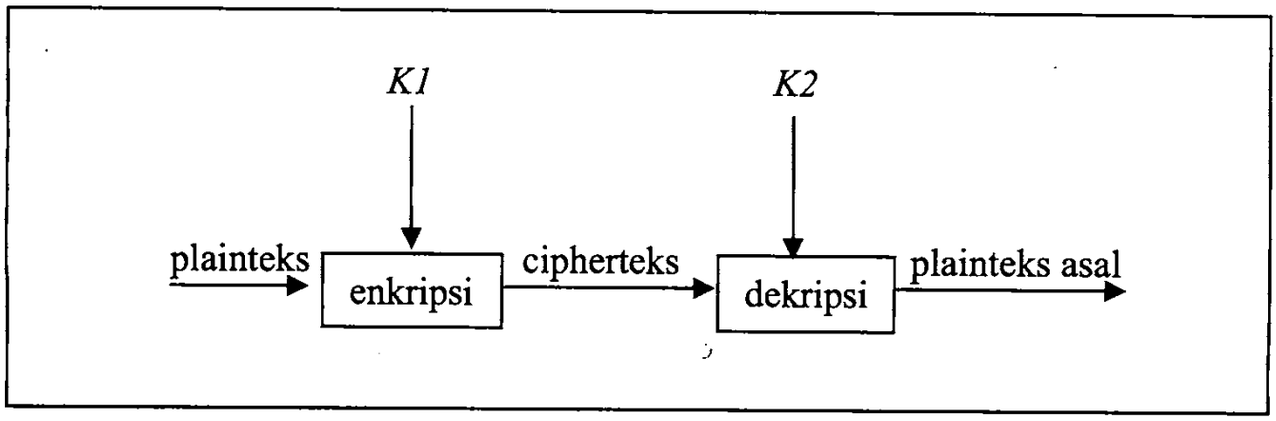
dan

$$D_{K2}(C) = P \qquad (5.15)$$

Kedua fungsi ini memenuhi

$$D_{K2}(E_{K1}(P)) = P$$

Gambar 5.3 memperlihatkan diagram proses enkripsi dan dekripsi pada kriptografi yang menggunakan kunci.



Gambar 5.3. Enkripsi dan Dekripsi pada algoritma kriptografi modern.

Jika *K1* = *K2* (yaitu, kunci untuk proses enkripsi sama dengan kunci untuk dekripsi), maka sistem kriptografinya dinamakan **sistem kriptografi kunci-simetri** (*symmetric-key cryptosystem*), (yang dimaksud dengan sistem kriptografi adalah gabungan dari algoritma kriptografi, kunci, plainteks, dan cipherteks) dan algoritma kriptografinya disebut **algoritma simetri**. Contoh algoritma simetri adalah *DES* (*Data Encyption Standard*). Sebaliknya, jika *K1* ≠ *K2* (yaitu, kunci untuk proses enkripsi berbeda dengan kunci untuk dekripsi), sistem kriptografinya dinamakan **sistem kriptografi nirsimetri** (*asymmetric cryptosystem*), dan algoritma kriptografinya disebut **algoritma nirsimetri**. Contoh algoritma nirsimetri adalah *RSA* (singkatan dari tiga nama penemu algoritmanya: Rivest-Shamir-Adleman).

Kriptografi simetri kadang-kadang disebut juga **kriptografi kunci-pribadi** (*private-key cryptography*) karena kunci enkripsi dan dekripsi sama dan harus dirahasiakan. Kelemahan dari sistem ini adalah baik pengirim maupun penerima pesan harus memiliki kunci yang sama, sehingga pengirim pesan harus mencari cara lain untuk memberitahukan kunci kepada penerima.

Kriptografi nirsimetri kadangkala disebut juga **kriptografi kunci-publik** (*public key cryptography*). Algoritma ini mempunyai dua buah kunci, yaitu kunci publik (*public key* – tidak rahasia) untuk enkripsi dan kunci pribadi (*secret key* – rahasia) untuk dekripsi. Pengirim pesan (*sender*) mengenkripsi pesan yang akan dikirim dengan menggunakan kunci publik si penerima pesan (*receiver*). Hanya penerima pesan yang dapat mendekripsi pesan karena hanya ia yang mengetahui kunci pribadi miliknya. Misalkan jaringan komputer menghubungkan komputer karyawan di kantor cabang dengan komputer menejer di kantor pusat. Seluruh karyawan diperintahkan bahwa kalau mereka mengirim laporan ke menejer di kantor pusat, mereka harus mengenkripsikan laporan tersebut dengan kunci publik milik menejer (jadi, kunci publik menejer diketahui oleh seluruh karyawan). Untuk mengeminversi data tersandi ke data asal, hanya menejer yang dapat melakukannya, karena dialah yang memegang kunci rahasia. Selama proses transmisi cipherteks dari kantor cabang ke kantor pusat melalui saluran

komunikasi mungkin saja data yang dikirim disadap oleh pihak ketiga, namun pihak ketiga ini tidak dapat mengeminversi cipherteks ke palinteksnya karena ia tidak mengetahui kunci untuk dekripsi.

DES (*Data Encryption Standard*)

DES memadukan teknik permutasi, ekspansi, kompaksi, dan substitusi, semuanya dilakukan dalam 16 kali perulangan. Panjang kunci *DES* adalah 8 karakter atau 64 bit. Dari 64 bit tersebut, hanya 56 bit saja yang dipakai dalam proses enkripsi. Tetapi patut dicatat bahwa dengan 56 bit itu akan terdapat 2^{56} atau 72.057.594.037.927.936 kemungkinan kunci. Jika orang yang tidak berhak mencoba keseluruhan kemungkinan kunci tersebut dengan menggunakan satu juta prosesor komputer yang bekerja secara paralel, maka dengan asumsi bahwa selama 1 detik dapat dicoba satu juta kemungkinan kunci, maka seluruh kemungkinan kunci tersebut memerlukan waktu 2284 tahun untuk menemukan kunci yang benar. Ini sebuah waktu yang lama, bahkan kriptanalis yang mencobanya pun sudah meninggal dunia sebelum waktu itu selesai. Algoritma *DES* tidak dibahas di dalam bab ini karena tidak relevan dengan pokok bahasan kita yang menyangkut penggunaan bilangan prima dan aritmetika modulo.

RSA (Rivest-Shamir-Adleman)

Algoritma *RSA* diperkenalkan oleh tiga peneliti dari *MIT* (*Massachussets Institute of Technology*), yaitu Ron Rivest, Adi Shamir, dan Len Adleman, pada tahun 1976. *RSA* mendasarkan proses enkripsi dan dekripsinya pada konsep bilangan prima dan aritmetika modulo. Baik kunci enkripsi maupun kunci dekripsi keduanya berupa bilangan bulat. Kunci enkripsi tidak dirahasiakan dan diketahui umum (sehingga dinamakan juga **kunci publik**), namun kunci untuk dekripsi bersifat rahasia. Kunci dekripsi dibangkitkan dari beberapa buah bilangan prima bersama-sama dengan kunci enkripsi. Untuk menemukan kunci dekripsi, orang harus memfaktorkan suatu bilangan non prima menjadi faktor primanya. Kenyataannya, memfaktorkan bilangan non prima menjadi faktor primannya bukanlah pekerjaan yang mudah. Belum ada algoritma yang mangkus (efisien) yang ditemukan untuk pemfaktoran itu. Semakin besar bilangan non primanya tentu semkain sulit pula pemfaktorannya. Semakin sulit pemfaktorannya, semakin kuat pula algoritma *RSA*. Algoritma *RSA* sebenarnya sederhana sekali. Secara ringkas, algoritma *RSA* terdiri dari tiga bagian, yaitu bagian untuk membangkitkan pasangan kunci, bagian untuk enkripsi, dan bagian untuk dekripsi:

ALGORITMA RSA

Pembangkitan pasangan kunci

1. Pilih dua buah bilangan prima sembarang, sebut a dan b. Jaga kerahasiaan a dan b ini.
2. Hitung $n = a\,b$. Besaran n tidak perlu dirahasiakan.
3. Hitung $m = (a - 1)(b - 1)$. Sekali m telah dihitung, a dan b dapat dihapus untuk mencegah diketahuinya oleh pihak lain.
4. Pilih sebuah bilangan bulat untuk kunci publik, sebut namanya e, yang relatif prima terhadap m.
5. Hitung kunci dekripsi, d, dengan kekongruenan $ed \equiv 1$ (**mod** m).

Enkripsi

1. Nyatakan pesan menjadi blok-blok plainteks: p_1, p_2, p_3, ... (harus dipenuhi persyaratan bahwa nilai p_i harus terletak dalam himpunan nilai 0, 1, 2, ..., $n - 1$ untuk menjamin hasil perhitungan tidak berada di luar himpunan)
2. Hitung blok cipherteks c_i untuk blok plainteks p_i dengan persamaan

$$c_i = p_i^e \textbf{ mod } n$$

yang dalam hal ini, e adalah kunci publik.

Dekripsi

1. Proses dekripsi dilakukan dengan menggunakan persamaan

$$p_i = c_i^d \textbf{ mod } n,$$

yang dalam hal ini, d adalah kunci pribadi.

Algoritma 5.7 Algoritma RSA

Perhatikan langkah 5 pada proses pembakitan pasangan kunci. Kekongruenan $ed \equiv 1$ (**mod** m) sama dengan ed **mod** $m = 1$. Menurut persamaan (5.6) yang menyatakan bahwa $a \equiv b$ (**mod** m) ekivalen dengan $a = b + km$, maka $ed \equiv 1$ (**mod** m) ekivalen dengan $ed = 1 + km$, sehingga d dapat dihitung dengan cara yang sederhana dengan persamaan

$$d = \frac{1 + km}{e} \qquad (5.16)$$

Dalam implementasi yang sebenarnya, nilai a dan b disarankan nilai yang sangat besar (100 angka) agar pekerjaan memfaktorkan n menjadi faktor primanya menjadi sangat sukar bahkan hampir tidak mungkin dapat dilakukan.

Contoh 5.19

Kita akan mengenkripsi pesan dengan algoritma *RSA*. Mula-mula, bangkitkan sepasang kunci. Sebagai ilustrasi, pilih $a = 47$ dan $b = 71$ (dalam praktek, a dan b harus bilangan yang besar), maka dapat dihitung nilai $n = ab = 3337$ dan $m = (a - 1)(b - 1) = 3220$. 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 seperti yang dituliskan pada langkah instruksi 4,

$$e\,d \equiv 1 \pmod{m}$$

Dengan menggunakan (5.16) kita menghitung kunci dekripsi d sebagai berikut:

$$d = \frac{1 + (k \times 3220)}{79}$$

Dengan mencoba nilai-nilai $k = 1, 2, 3, \ldots$, diperoleh nilai d yang bulat adalah 1019. Ini adalah kunci dekripsi yang harus dirahasiakan.

Misalkan plainteks yang akan dienkripsikan adalah P = HARI INI (atau dalam desimal ASCII-nya adalah 7265827332737873). Pecah P menjadi blok yang lebih kecil, misalnya P dipecah menjadi enam blok yang berukuran 3 digit:

$p_1 = 726$ $\quad$ $p_4 = 273$
$p_2 = 582$ $\quad$ $p_5 = 787$
$p_3 = 733$ $\quad$ $p_6 = 003$

Nilai-nilai p_i ini masih terletak di dalam rentang nilai 0 sampai 3337 – 1. Blok pertama dienkripsikan sebagai

$$726^{79} \textbf{ mod } 3337 = 1{,}430456768828466034712340994000 7 \cdot 10^{226} \bmod 337 = 215 = c_1$$

Blok kedua dienkripsikan sebagai

$$582^{79} \textbf{ mod } 3337 = 776 = c_2$$

Dengan melakukan proses yang sama untuk sisa blok lainnya, dihasilkan cipherteks C = 215 776 1743 933 1731 158.

Proses dekripsi dilakukan dengan menggunakan kunci rahasia d = 1019, jadi, blok c_1 didekripsikan sebagai

$$215^{1019} \textbf{ mod } 3337 = 726 = p_1$$

Blok c_2 didekripsikan sebagai

$$776^{1019} \bmod 3337 = 582 = p_2$$

Blok plainteks yang lain dikembalikan dengan cara yang serupa. Akhirnya kita memperoleh kembali plainteks semula P = 7265827332737873 atau dalam bentuk karakter adalah P = HARI INI. ■

Perhitungan perpangkatan pada proses enkripsi ($c_i = p_i^e$ **mod** n)dan dekripsi ($p_i = c_i^d$ **mod** n) membutuhkan bilangan yang sangat besar. Untuk menghindari penggunaan bilangan yang besar, maka dapat digunakan penyederhanaan dengan persamaan berikut:

$$ab \textbf{ mod } m = [(a \textbf{ mod } m)(b \textbf{ mod } m)] \textbf{ mod } m \qquad (5.17)$$

Contoh 5.20

Sebagai ilustrasi, untuk menghitung 572^{37} mod 713 dapat digunakan manipulasi dengan persamaan 3.14 sebagai berikut:

$572^{37} = 572^{32} \cdot 572^4 \cdot 572$

572^2 **mod** 713 = 327184 **mod** 713 = 630

572^4 **mod** 713 = $572^2 \cdot 572^2$ **mod** 713 = [(572^2 **mod** 713)(572^2 **mod** 713)] **mod** 713
= 630^2 **mod** 713 = 396900 **mod** 713 = 472

572^8 **mod** 713 = $572^4 \cdot 572^4$ **mod** 713 = [(572^4 **mod** 713)(572^4 **mod** 713)] **mod** 713
= 472^2 **mod** 713 = 222784 **mod** 713 = 328

572^{16} **mod** 713 = $572^8 \cdot 572^8$ **mod** 713 = [(572^8 **mod** 713)(572^8 **mod** 713)] **mod** 713
= 328^2 **mod** 713 = 107584 **mod** 713 = 634

572^{32} **mod** 713 = $572^{16} \cdot 572^{16}$ mod 713 = [(572^{16} **mod** 713)(572^{16} **mod** 713)] **mod** 713
= 634^2 **mod** 713 = 401956 **mod** 713 = 537

572^{36} **mod** 713 = $572^{32} \cdot 572^4$ **mod** 713 = [(572^{32} **mod** 713)(572^4 **mod** 713)] **mod** 713
= 537 . 472 **mod** 713 = 253464 **mod** 713 = 349

572^{37} **mod** 713 = $572^{36} \cdot 572$ **mod** 713 = [(572^{36} **mod** 713)(572 **mod** 713)] **mod** 713
= 349 . 572 **mod** 713 = 199628 **mod** 713 = 701

Jadi, 572^{37} **mod** 713 = 701

■

Kekuatan dan Keamanan RSA

Seperti yang sudah dikatakan sebelumnya, kekuatan algoritma *RSA* terletak pada tingkat kesulitan dalam memfaktorkan bilangan menjadi faktor primanya, yang dalam hal ini adalah memfaktorkan n menjadi a dan b. Sekali n berhasil difaktorkan menjadi a dan b, maka $m = (a - 1)(b - 1)$ dapat dihitung. Selanjutnya, karena kunci enkrispi e diumumkan (tidak rahasia), maka kunci dekripsi d dapat dihitung dari persamaan $ed \equiv 1$ (**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 $n = ab$ akan berukuran lebih dari 200 digit. Bayangkanlah berapa besar usaha kerja yang diperlukan untuk memfaktorkan bilangan bulat 200 digit menjadi faktor primanya. Menurut Rivest dan kawan-

kawan, usaha 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).

Untunglah algoritma yang paling mangkus untuk memfaktorkan bilangan yang besar belum ditemukan. Inilah yang membuat algoritma *RSA* tetap dipakai hingga saat ini. Selagi belum ditemukan algoritma yang mangkus untuk memfaktorkan bilangan bulat menjadi faktor primanya, maka algoritma *RSA* masih direkomendasikan untuk penyandian pesan.

5.11 Fungsi *Hash*

Data yang disimpan di dalam memori komputer perlu ditempatkan dalam suatu cara sedemikian sehingga pencariannya dapat dilakukan dengan cepat. Setiap data yang berupa record mempunyai *field* **kunci** yang unik yang membedakan suatu *record* dengan *record* lainnya. Fungsi *hash* (*hash function*) digunakan untuk menempatkan suatu *record* yang mempunyai nilai kunci k. Fungsi *hash* yang paling umum berbentuk

$$h(k) = k \bmod m \qquad (5.18)$$

yang dalam hal ini m adalah jumlah lokasi memori yang tersedia (misalkan memori berbentuk sel-sel yang diberi indeks 0 sampai $m - 1$). Fungsi h di atas menempatkan *record* dengan kunci k pada suatu lokasi memori yang beralamat $h(k)$.

Andaikan $m = 11$, sehingga kita mempunyai sel-sel memori yang diberi indeks 0 sampai 10. Kita akan menyimpan data *record* yang masing-masing mempunyai kunci 15, 558, 32, 132, 102, dan 5 [JOH97]. Pada mulanya sel-sel meori dalam keadaan kosong.

Keenam data *record* tersebut masing-masing disimpan pada lokasi yang dihitung sebagai berikut:

$h(15) = 15 \bmod 11 = 4$
$h(558) = 558 \bmod 11 = 8$
$h(32) = 32 \bmod 11 = 10$
$h(132) = 132 \bmod 11 = 0$
$h(102) = 102 \bmod 11 = 3$
$h(5) = 5 \bmod 11 = 5$

Keadaan sel-sel memori setelah penyimpanan keenam data *record* tersebut digambarkan seperti berikut ini:

132			102	15	5			558		32
0	1	2	3	4	5	6	7	8	9	10

Karena fungsi *hash* bukanlah fungsi satu-ke-satu (beberapa nilai k yang berbeda dapat menghasilkan nilai $h(k)$ yang sama), maka dapat terjadi **bentrokan** (*collision*) dalam penempatan suatu data *record*. Misalnya kita akan menempatkan data record dengan kunci 257. Perhitungan *hash* menghasilkan

$$h(257) = 257 \textbf{ mod } 11 = 4$$

padahal sel memori dengan lokasi 4 sudah terisi. Kita katakan telah terjadi bentrokan. Untuk mengatasi bentrokan perlu diterapkan **kebijakan resolusi bentrokan** (*collision resolution policy*). Satu kebijakan resolusi bentrokan adalah mencari sel tak terisi tertinggi berikutnya (dengan 0 diasumsikan mengikuti 10). Jika kita terapkan kebijakan ini, maka data *record* dengan kunci 257 ditempatkan pada lokasi 6.

Jika kita ingin mencari data *record* tertentu, maka kita gunakan fungsi hash kembali. Misalkan kita akan mencari data record dengan kunci p, maka kita hitung $h(p) = p$ **mod** 11, misalkan $h(p) = q$. Jika *record p* sama dengan isi sel pada lokasi q, kita katakan lokasi *record p* ditemukan. Sebaliknya, jika *record p* tidak sama dengan isi sel pada lokasi q, maka kita melihat pada posisi tertinggi berikutnya (sekali lagi, 0 diasumsikan mengikuti 10); jika *record p* tidak berada pada posisi ini, kita lihat lagi pada posisi berikutnya, demikian seterusnya. Jika kita mencapai sel kosong atau kembali ke posisi semula, kita simpulkan bahwa *record p* tidak ada.

5.12 *International Standard Book Number* (ISBN)

Buku-buku yang diterbitkan oleh penerbit resmi selalu disertai dengan kode *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, kode penerbit, kode yang diberikan secara unik kepada buku tersebut, dan sebuah karakter uji (dapat berupa angka atau huruf X untuk merepresentasikan angka 10). Karakter uji digunakan untuk mevalidasi *ISBN*, tepatnya untuk mendeteksi kesalahan pada karakter *ISBN* atau kesalahan karena perpindahan angka-angkanya. Karakter uji dipilih sedemikian sehingga

$$\sum_{i=1}^{10} ix_i \equiv 0 \ (mod \ 11)$$

yang dalam hal ini x_i adalah karakter yang ke-i di dalam kode *ISBN*. Untuk mendapatkan karakter uji, kita cukup menghitung

$$\sum_{i=1}^{9} ix_i \ \textbf{mod}\ 11 = \text{karakter uji} \qquad (5.17)$$

Untuk kode *ISBN* 0–3015–4561–8, 0 adalah kode kelompok negara berbahasa Inggris, 3015 adalah kode penerbit, 4561 adalah kode unik untuk buku yang diterbitkan oleh penerbit tersebut, dan 8 adalah karakter uji. Karakter uji ini didapatkan sebagai berikut:

$$1 \cdot 0 + 2 \cdot 3 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 5 + 6 \cdot 4 + 7 \cdot 5 + 8 \cdot 6 + 9 \cdot 1 = 151$$

Jadi, karakter ujinya adalah 151 **mod** 11 = 8. Catatlah bahwa untuk kode *ISBN* ini,

$$\sum_{i=1}^{10} ix_i = \sum_{i=1}^{9} ix_i + 10x_{10} = 151 + 10 \cdot 8 = 231$$

dan 231 **mod** 11 = 0 atau 231 ≡ 0 (**mod** 11).

Contoh 5.20

Nomor *ISBN* sebuah buku terbitan penerbit Indonesia adalah 979-939p-04-5. Tentukan p.

Penyelesaian:
Diketahui karakter uji *ISBN* = 5. Ini berarti

$$\sum_{i=1}^{9} ix_i \ \textbf{mod}\ 11 = 5$$

Mula-mula hitung

$$\sum_{i=1}^{9} ix_i = 1 \cdot 9 + 2 \cdot 7 + 3 \cdot 9 + 4 \cdot 9 + 5 \cdot 3 + 6 \cdot 9 + 7 \cdot p + 8 \cdot 0 + 9 \cdot 4$$

$$= 9 + 14 + 27 + 36 + 15 + 54 + 7p + 0 + 36 = 191 + 7p$$

Jadi,

$$(191 + 7x) \ \textbf{mod}\ 11 = 5$$

atau

$$p = \frac{11k + 5 - 191}{7} = \frac{11k - 186}{7}$$

Nilai-nilai k yang menghasilkan x bulat adalah k = …, –6, 1, 8, 15, 22, 28, … . Agar *ISBN* sah maka p haruslah memenuhi $0 \geq p \geq 9$. Untuk $k = 22$ didapatkan $p = 8$. ■

5.13 Pembangkit Bilangan Acak Semu

Bilangan acak (*random*) banyak digunakan di dalam program komputer, misalnya untuk program simulasi (misalnya mensimulasikan waktu kedatangan nasabah di bank, pompa bensin, dan sebaiknya), program kriptografi, aplikasi statistik, dan sebagainya.

Tidak ada komputasi yang benar-benar menghasilkan deret bilangan acak secara sempurna. Bilangan acak yang dihasilkan dengan rumus-rumus matematika adalah bilangan acak semu (*pseudo*), karena pembangkitan bilangannya dapat diulang kembali. Pembangkit deret bilangan acak semacam itu disebut **pembangkit bilangan acak semu** (*pseudo-random number generator* atau *PRNG*).

Metode yang paling umum digunakan untuk membangkitkan bilangan acak adalah dengan pembangkit bilangan acak kongruen-lanjar (*linear congruential generator* atau *LCG*) adalah *PRNG* yang berbentuk:

$$x_n = (ax_{n-1} + b) \textbf{ mod } m \qquad (5.18)$$

yang dalam hal ini,

x_n = bilangan acak ke-n dari deretnya
x_{n-1} = bilangan acak sebelumnya
a = faktor pengali
b = *increment*
m = modulus
(a, b, dan m semuanya konstanta)

Kunci pembangkit adalah x_0 yang disebut **umpan** (*seed*).

LCG mempunyai periode tidak lebih besar dari m. Jika a, b, dan m dipilih secara tepat (misalnya b seharusnya relatif prima terhadap m), maka *LCG* akan mempunyai periode maksimal, yaitu $m - 1$.

Contoh 5.21

Bangkitkan bilangan acak dengan menggunakan *LCG*, $m = 17$, $a = 7$, $b = 11$, dan $x_0 = 0$.

Penyelesaian:
Persamaan *LCG* berbentuk

$$x_n = (7x_{n-1} + 11) \textbf{ mod } 17$$

Lakukan perhitungan sebagai berikut:

$x_1 = (7x_0 + 11)$ **mod** $17 = (7 \cdot 0 + 11)$ **mod** $17 = 11$ **mod** $17 = 11$
$x_2 = (7x_1 + 11)$ **mod** $17 = (7 \cdot 11 + 11)$ **mod** $17 = 88$ **mod** $17 = 3$
dst...

Hasil perhitungan disajikan dalam bentuk tabel seperti di bawah ini:

n	x_n
0	0
1	11
2	3
3	15
4	14
5	7
6	9
7	6
8	2
9	8
10	16
11	4
12	5
13	12
14	10
15	13
16	*0*
17	*11*
18	*3*
19	*15*
20	*14*
21	*7*
22	*9*
23	*6*
24	*2*

Pada $n = 16$, nilai $x_{16} = x_0$, maka bilangan acak berikutnya (x_{17}, x_{18}, dst) akan berulang kembali. Inilah alasannya mengapa *LCG* termasuk ke dalam pembangkit bilangan acak semu. ■

5.14 Ragam Soal dan Penyelesaian

Contoh 5.22

(i) Berapa –211 **mod** 11?
(ii) Misalkan $m = -101$ dan $n = 13$. Nyatakan m dan n dalam $m = nq + r$

Penyelesaian:
(i) -211 **mod** $11 = 11 - (|-211|$ **mod** $11) = 11 - 2 = 9$
(ii) Ada banyak cara untuk menyelesaikan soal ini, salahs atunya adalah sebagi berikut:

$$\begin{aligned} -101 &= 13\,q + r \\ r &= -101 \textbf{ mod } 13 \\ &= 13 - (|-101| \bmod 13) \\ &= 13 - 10 \\ &= 3 \end{aligned}$$

$$\begin{aligned} -101 &= 13\,q + 3 \\ q &= (-101 - 3)\,)\,13 \\ &= -8 \end{aligned}$$

Jadi, $-101 = 13\,(-8) + 3$ ■

Contoh 5.23

Nyatakan PBB dari 172 dan 324 sebagai kombinasi lanjar dari bilangan-bilangan tersebut.

Penyelesaian:
Lakukan pembagian dengan menggunakan algoritma Euclidan untuk mendapatkan PBB(172, 324):

$$\begin{aligned} 324 &= 1\,(172) + 152 && \text{(i)} \\ 172 &= 1\,(152) + 20 && \text{(ii)} \\ 152 &= 7\,(20) + 12 && \text{(iii)} \\ 20 &= 1\,(12) + 8 && \text{(iv)} \\ 12 &= 1\,(8) + 4 && \text{(v)} \\ 8 &= 1\,(4) + 0 && \text{(vi)} \end{aligned}$$

Sisa pembagian terakhir sebelum 0 adalah 4, maka PBB(324, 172) = 4. Lakukan proses substitusi dari persamaan–persamaan di atas:

$$\begin{aligned} 4 &= 12 - 1\,(8) \\ &= 12 - (20 - 12) \\ &= -20 + 2\,(12) \\ &= -20 + 2\,(152 - 7\,(20)) \\ &= 2\,(152) + (-15)\,(20) \\ &= 2\,(152) + (-15)\,(172 - 152) \\ &= -15\,(172) + (17)\,(152) \\ &= -15\,(172) + (17)\,(324 - 172) \\ &= 17\,(324) + (-32)\,(172) \end{aligned}$$

Jadi, PBB(324, 172) dapat dinyatakan sebagai PBB dari 324 dan 172 sebagai:

$$4 = 17\,(324) + (-32)\,(172)$$ ■

Contoh 5.24

Buktikan bahwa jika a, b, dan m adalah bilangan bulat sedemikian sehingga $m \geq 2$, dan $a \equiv b$ (**mod** m) dan $c \equiv d$ (**mod** m), maka $ac \equiv bd$ (**mod** m).

Penyelesaian:

$a \equiv b$ (**mod** m) $\Leftrightarrow a = b + k_1 m$

$c \equiv d$ (**mod** m) $\Leftrightarrow c = d + k_2 m$

$ac = (b + k_1 m)(d + k_2 m)$

$= bd + bk_2 m + dk_1 m + k_1 k_2 m^2$

$= bd + (bk_2 + dk_1 + k_1 k_2 m)\, m$

$ac = bd + Km$ dengan $K = bk_2 + dk_1 + k_1 k_2 m$

$= bd + Km \Leftrightarrow ac \equiv bd$ (**mod** m) ■

Contoh 5.25

Misalkan m adalah bilangan bulat positif. Buktikan bahwa jika a **mod** $m = b$ **mod** m, maka $a \equiv b$ (**mod** m).

Penyelesaian:

a **mod** $m = b$ **mod** m

$a + k_1 m = b + k_2 . m$

$a = b + (k_2 - k_1)\, m$

$a - b = (k_2 - k_1)\, m$

Persamaan yang terakhir mengimplikasikan bahwa m habis membagi $(a - b)$, sehingga kita dapat menyatakan bahwa $a \equiv b$ (**mod** m). ■

Contoh 5.26

Sebuah area parkir mempunyai sejumlah *slot* atau *space* yang dinomori 0 sampai 30. Mobil yang hendak parkir di area tersebut ditentukan dengan sebuah fungsi *hash*. Fungsi *hash* tersebut menentukan nomor *slot* yang akan ditempati mobil yang hendak parkir berdasarkan 3 angka terakhir pada plat nomor polisinya.

(a) Tentukan fungsi *hash* yang dimaksudkan.

(b) Tentukan nomor *slot* yang ditempati mobil yang datang berturut-turut dengan 3 angka terakhir pada plat nomor polisisnya adalah 327, 100, 121, 310, 414, 110, 017

Penyelesaian:

(a) Fungsi *hash* : $h(x) = x$ **mod** 31

(b) Nomor slot yang ditempati mobil dihitung dengan fungsi *hash* (a) di atas:

$h(327) = 327$ **mod** $31 = 17$

$h(100) = 100$ **mod** $31 = 7$

$h(121) = 121$ **mod** $31 = 28$

$h(310) = 310$ **mod** $31 = 0$

$h(414) = 414$ **mod** $31 = 11$

$h(110) = 110$ **mod** $31 = 17$, karena *slot* sudah terisi maka isi *slot* kosong berikutnya, yaitu 18

$h(017) = 017$ **mod** $31 = 17$, karena *slot* sudah terisi maka 18, tetapi karena 18 sudah terisi, maka isi *slot* berikutnya, yaitu 19 ■

Contoh 5.27

Nomor *ISBN* sebuah buku yang terbaca oleh *bar code* di toko buku adalah 0-07-053965-X. Apakah nomor *ISBN* tersebut sah? Jika tidak, bagaimana seharusnya?

Penyelesaian:
Hal pertama yang harus dilakukan adalah memeriksa apakah benar karakter uji *ISBN* tersebut adalah *X* (representasi angka 10) dengan perhitungan berikut:

$$\sum_{i=1}^{9} ix_i \textbf{ mod } 11 = \text{karakter uji}$$

dalam hal ini,

$$\begin{aligned}\sum_{i=1}^{9} ix_i &= 1.0 + 2.0 + 3.7 + 4.0 + 5.5 + 6.3 + 7.9 + 8.6 + 9.5 \\ &= 21 + 25 + 18 + 63 + 48 + 45 \\ &= 220\end{aligned}$$

sehingga

$$\sum_{i=1}^{9} ix_i \textbf{ mod } 11 = 220 \textbf{ mod } 11 = 0$$

Hasil perhitungan yang terakhir adalah 0 (bukan 10) sehingga menunjukkan bahwa *X* tidak memenuhi sifat karakter uji nomor ISBN di atas. Oleh sebab itu, karakter uji *ISBN* di atas seharusnya bukan X, melainkan 0. Dengan demikian, nomor *ISBN* yang tepat adalah 0-07-053965-0.

Soal Latihan

1. Apakah 19 habis membagi bilangan bulat berikut:
   (a) 89 (b) 561 (c) 209 (d) 773 (e) 8721

2. Carilah bilangan bulat q dan r sehingga $m = nq + r$
   (a) $m = 45, n = 6$ (c) $m = 106, n = 12$ (e) $m = -221, n = 12$
   (b) $m = 66, n = 11$ (d) $m = 0, n = 47$ (f) $m = -246, n = 49$

3. Perlihatkan bahwa jika $p \mid q$ dan $r \mid s$, maka $pq \mid rs$.

4. Misalkan m, n, dan c adalah bilangan bulat. Tunjukkan bahwa jika c adalah pembagi bersama terbesar dari m dan n, maka $c \mid (m - n)$.

5. Perlihatkan bahwa jika p, q, dan r bilangan bulat sedemikian sehingga $pr \mid qr$, maka $p \mid q$.

6. Hitung hasil pembagian modulo berikut:
   (a) -173 **mod** 21 (b) -340 **mod** 9
   (c) 0 **mod** 34 (d) -9821 **mod** 45

7. Jika m bilangan bulat positif, perlihatkan bahwa a **mod** $m \equiv b$ **mod** m jika $a \equiv b$ (**mod** m).

8. Perlihatkan bahwa jika $a \equiv b$ (**mod** m) dan $c \equiv d$ (**mod** m), yang dalam hal ini a, b, c, d, dan m adalah bilangan bulat, maka $a - c \equiv (b - d)$ **mod** m.

9. Buktikan bahwa jika a, b, k, dan m adalah bilangan bulat sedemikian sehingga $k \geq 1$, $m \geq 2$, dan $a \equiv b$ (**mod** m), maka $a^k \equiv b^k$ (**mod** m).

10. Misalkan a, dan b bilangan bulat dan m dan n adalah bilangan bulat positif lebih besar dari 1. Buktikan bahwa jika $n \mid m$ dan $a \equiv b$ (**mod** m), maka $a \equiv b$ (**mod** n).

11. Tunjukkan bahwa jika a, b, dan m bilangan bulat sedemikian sehingga $m \geq 2$, dan $a \equiv b$ (**mod** m), maka PBB(a, m) = PBB(b, m).

12. Tentukan PBB dari pasangan bilangan bulat a dan b berikut:
   (a) 220, 1400 (b) 315, 825
   (c) 110, 273 (d) 2475, 32670 (e) -456, 688

13. Tentukan inversi (*invers*) dari a modulo m jika $a = -39$ dan $m = 14$.

14. Nyatakan PBB dari soal nomor 7 di atas dalam bentuk kombinasi lanjar $ma + nb$.

15. Tuliskan 5 buah bilangan bulat yang kongruen dengan 4 modulo 12.

16. Tentukan pasangan bilangan bulat yang relatif prima satu sama lain:
    (a) 21, 34, 55 (b) 25, 41, 49, 64 (c) 17, 18, 19, 23

17. Andaikan bahwa a dan b bilangan bulat positif. Tunjukkan bahwa PBB(a, b) = PBB(a, $a + b$).

18. Pecahkan kekongruenan lanjar berikut:
    (a) $4x \equiv 5$ (**mod** 8) (b) $2x \equiv 7$ (**mod** 17) (c) $5x \equiv 10$ (**mod** 12)

19. Tentukan inversi *(invers)* dari a modulo m berikut:
    (a) $a = 34$, $m = 5$ (b) $a = 178$, $m = 62$ (c) $a = -341$, $m = 17$

20. Tunjukkan bahwa $2^{340} \equiv 1$ (**mod** 11) dengan Teorema Fermat dan memperhatikan bahwa $2^{340} = (2^{10})^{34}$.

21. Misalkan Julius Caesar mengenkripsikan pesan dengan cara menggeser huruf abjad 8 posisi ke kanan.
    (a) Nyatakan fungsi matematik yang memetakan plainteks ke cipherteks dengan metode di atas.
    (b) Misalkan $A = 0$, $B = 1$, ..., $Z = 25$. Tentukan cipherteks dari pesan "DI RUMAH" dengan fungsi tsb.

22. Enkripsikan pesan HELLO WORLD dengan algoritma *RSA* dan menggunakan nilai-nilai $a = 23$, $b = 31$, dan $e = 29$.

23. Sembilan angka pertama dari kode *ISBN* sebuah buku adalah 0–07–053965. Tentukan karakter uji untuk buku ini.

24. *ISBN* sebuah buku mengenai algoritma adalah 0–201–57p859–1, yang dalam hal ini p adalah angka. Berapa nilai p?

25. Tunjukkan bagaimana sekumpulan data dengan kunci-kunci sebagai berikut: 714, 631, 26, 373, 775, 906, 509, 2032, 42, 4, 136, 1028 ditempatkan di dalam memori dengan fungsi *hash* $h(k) = k$ **mod** 17.

26. Tentukan bilangan acak yang dihasilkan oleh $x_{n+1} = (4x_n + 1)$ **mod** 7 dengan umpan $x_0 = 7$.

27. Tentuka solusi dari sistem kekongruenan berikut: $x \equiv 5$ (**mod** 6), $x \equiv 3$ (**mod** 10), $x \equiv 8$ (**mod** 13).