Aplikasi Teori Bilangan Bulat dalam Pembangkitan Bilangan Acak Semu

Ferdian Thung 13507127
Program Studi Teknik Informatika ITB, Jalan Ganesha 10 Bandung, Jawa Barat,
email: if17127@students.if.itb.ac.id

Abstrak – *Makalah ini membahas metode-metode pembangkitan bilangan acak semu melalui penerapan teori bilangan bulat. Bilangan acak semu merupakan sebuah bilangan yang dihasilkan melalui suatu proses komputasi atau algoritma tertentu sehingga keluarannya tampak acak dari luar. Pada makalah ini akan dibahas beberapa pseudorandom number generator yang menggunakan prinsip aritmatika modulo dalam pembangkitan bilangan acak semu. Bilangan semacam ini telah banyak diaplikasikan dalam games, kriptografi, statistik, serta bidang lain yang membutuhkan bilangan acak sebagai variabel inputnya.*

Kata Kunci*: bilangan acak semu, aritmatika modulo, pseudorandom number generator, linear congruential generator, Blum Blum Shut*

1. PENDAHULUAN

1.1 Teori Bilangan Bulat

[1]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. Bilangan bulat memiliki sifat pembagian sebagai berikut :

- Misalkan a dan b adalah dua buah bilangan bulat dengan syarat $a \neq 0$. Kita menyatakan bahwa a habis membagi b jika terdapat bilangan bulat c sedemikian sehingga $b = ac$.
- Notasi: $a \mid b$ jika $b = ac$, $c \in \mathbf{Z}$ dan $a \neq 0$. ($\mathbf{Z}$ = himpunan bilangan bulat).

Adapun beberapa teori dan pengertian dasar yang penting dalam teori bilangan bulat, yaitu :

1. Modulo
   Misalkan a adalah bilangan bulat dan m adalah bilangan bulat lebih besar dari nol. Operasi a mod m memberikan sisa jika a dibagi dengan m. Pernyataan ini dinotasikan dalam persamaan $a \bmod m = r$ sedemikian sehingga $a = mq + r$, dengan $0 \leq r < m$. Bilangan m disebut modulus atau modulo, dan hasil aritmetika modulo m terletak di dalam himpunan $\{0, 1, 2, \ldots, m-1\}$
2. Pembagi Bersama Terbesar
   Misalkan a dan b adalah bilangan bulat tidak nol. Pembagi bersama terbesar dari a dan b adalah bilangan bulat terbesar d sedemikian sehingga d | a dan d | b. Dalam hal ini, kita nyatakan bahwa PBB(a,b) = d
3. Kongruen
   Misalkan a dan b bilangan bulat dan m adalah bilangan lebih besar dari nol, maka $a \equiv b \pmod m$ jika m habis membagi $a - b$. Sebaliknya, jika a tidak kongruen dengan b dalam modulus m, maka ditulis $a \not\equiv b \pmod m$.
4. Balikan Modulo (Modulo Invers)
   Jika a dan m relatif prima dan m lebih besar dari satu, maka kita dapat menemukan balikan (*invers*) dari a modulo m. Balikan dari a modulo m adalah bilangan bulat $\overline{a}$ sedemikian sehingga:
   $$a\overline{a} \equiv 1 \pmod m$$

1.2 Bilangan Acak Semu

Bilangan acak berarti suatu bilangan yang diambil dari sekumpulan bilangan di mana tiap-tiap elemen dari kumpulan bilangan ini mempunyai peluang yang sama untuk terambil . Misalkan kumpulan bilangan berjumlah n, maka masing-masing elemen akan mempunyai peluang terambil sebesar 1/n. Jika jumlah bilangan yang akan diambil lebih dari satu, maka masing - masing proses pengambilannya harus bersifat bebas secara statistik (*statistically independent*).

Manusia sudah mengenal bilangan acak selama ribuan tahun, misalnya dalam permainan yang berkaitan dengan peluang. Contoh yang paling sederhana adalah dadu. Banyak orang sudah memiliki pemahaman intuitif bahwa setiap sisi dadu akan muncul sekitar 1/6 kali. Dari pemahaman intuitif tersebut, bilangan acak dapat didefinisikan sebagai deretan bilangan-bilangan yang tidak dapat diprediksi sebelum bilangan tersebut dibangkitkan.

Sayangnya, bilangan yang benar-benar acak sangatlah sulit untuk dibangkitkan, terutama pada komputer yang memang didesain deterministik. Hal ini menyebabkan bilangan acak yang

dihasilkan oleh komputer disebut bilangan acak semu (*pseudorandom number*).

Bilangan acak semu ini sendiri memiliki perbedaan pengertian antara bilangan acak semu yang digunakan dalam konteks pemrograman biasa (misalnya untuk permainan, simulasi, dan statistik) dengan bilangan acak semu dalam konteks kriptografi. Dalam kriptografi, bilangan acak adalah bilangan yang tidak dapat diprediksi nilainya sebelum bilangan tersebut dihasilkan atau dibangkitkan. Secara umum, jika bilangan acak yang akan dihasilkan antara [0..n-1], bilangan tersebut tidak dapat diprediksi kemungkinan kemunculannya lebih dari 1/n.

2. PEMBANGKIT BILANGAN ACAK SEMU

Pseudorandom Number Generator (PNRG) atau pembangkit bilangan acak semu adalah sebuah algoritma yang membangkitkan sebuah deret bilangan yang tidak benar-benar acak. Keluaran dari pembangkit bilangan acak semu ini hanya mendekati beberapa sifat yang dimiliki bilangan acak.

Awal munculnya PRNG pada komputer diusulkan oleh John von Neumann pada tahun 1946 dan dikenal sebagai *middle-square method*. Metode ini sangat sederhana. Pertama-tama, dipilih bilangan sembarang dan bilangan tersebut dikuadratkan. Kemudian, beberapa digit tengah bilangan kuadrat yang dihasilkan tersebut diambil. Bilangan ini merupakan bilangan acak yang dihasilkan dan akan menjadi umpan untuk menghasilkan bilangan acak selanjutnya.

Sebagai contoh, dipilih bilangan 1234, kemudian dikuadratkan menjadi 1522756 atau dapat ditulis 01522756 dalam format 8 digit karena bilangan yang dipilih pertama adalah 4 digit. 5227 merupakan bilangan yang dihasilkan pada iterasi pertama sebagai bilangan acak. Iterasi selanjutnya menghasilkan 3215. Kelemahan dari metode ini adalah semua deret-deretnya akan dengan cepat mengulang dirinya sendiri.

Pada dasarnya, semua PRNG berjalan di atas sebuah komputer yang deterministik sehingga keluaran yang dihasilkannya akan memiliki sifat yang tidak dimiliki bilangan acak alami yaitu periode. Hal ini berarti pada putaran tertentu setelah dijalankan akan dihasilkan deret yang berulang. Hal ini dikarenakan sebuah pembangkit bilangan acak memiliki memori terbatas. Akibatnya, setelah beberapa waktu pembangkit tersebut akan kembali pada kondisi semula dan hal ini menyebabkan pengulangan deret yang dihasilkan sebelumnya. Pembangkit yang tidak memiliki periode *(non-periodic generator*) dapat saja dirancang pada sebuah komputer yang deterministik, tetapi dibutuhkan memori yang tidak terbatas selama program pembangkit bilangan dijalankan. Tentunya hal tersebut tidak mungkin dilakukan.

Akan tetapi, konsekuensi yang dihasilkan dari deterministik komputer pada prakteknya dapat saja dihindari. Panjang dari maksimum periode dibuat sepanjang mungkin sehingga tidak ada komputer yang dapat mencapai satu periode dalam waktu yang diharapkan. Jika satu periode tidak dicapai maka pengulangan deret bilangan acak tidak terjadi. Walau begitu, penggunaan cara seperti ini tidak cukup baik untuk beberapa aplikasi yang membutuhkan waktu komputasi yang cepat, karena semakin panjang suatu periode akan membutuhkan memori komputer yang besar pula.

Saat ini, bilangan acak semu banyak digunakan dalam berbagai bidang seperti untuk simulasi dalam ilmu fisika, matematika, biologi, dan sebagainya. Walaupun terlihat sederhana untuk mendapatkan bilangan acak, tetapi diperlukan analisis matematika yang teliti untuk membangkitkan bilangan seacak mungkin. Berikut ini beberapa pembangkit bilangan acak semu :

1. Linear Congruential Generators
2. Blum Blum Shub
3. Inversive Congruential Generator
4. Generalised Feedback Shift Registers
5. Lagged Fibonacci Generators
6. Mersenne Twister
7. Linear Feedback Shift Registers
8. Digital Inversive Congruential Generator

Dari contoh-contoh pembangkit bilangan semu di atas, tidak semuanya menggunakan prinsip aritmatika modulo untuk membangkitkan bilangan acak semu. Dua di antaranya yaitu Linear Congruential Generator (LCG) dan Blum Blum Shub akan dibahas secara lebih mendetail pada bab selanjutnya.

3. LINEAR CONGRUENTIAL GENERATOR

Linear Congruential Generator (LCG) adalah *PRNG* yang berbentuk:

(1)

$$X_{n+1} = (aX_n + b) \bmod m$$

di mana :

X_n = bilangan acak ke-(n+1) dari deretnya
X_n = bilangan acak sebelumnya
a = faktor pengali
b = increment
m = modulus

Persamaan (1) memiliki nilai awal X_0 sebagai kunci pembangkit atau sering juga disebut umpan (seed). LCG mempunyai periode tidak lebih besar dari m dan akan mempunyai periode penuh jika memenuhi syarat sebagai berikut:

1. b relatif prima terhadap m.
2. a-1 dapat dibagi dengan semua faktor prima dari m.
3. a-1 adalah kelipatan 4 jika m adalah kelipatan 4.
4. m > maks(a, b, X_0).
5. a > 0, b > 0.

Misalkan nilai-nilai untuk

$X1_n \rightarrow a = 5,\ b = 13, M = 23$ dan $X_0 = 0$

Maka akan didapat rumusan sebagai berikut:

$$X1_{n+1} = (5X1_n + 13) \bmod 23$$

Kemudian kita hitung nilai Xn seperti dapat dilihat pada tabel 1.

n	X_n
0	0
1	13
2	9
3	12
4	4
5	10
6	17
7	6
8	20
9	21
10	3
11	5
12	15
13	19
14	16
15	1
16	18
17	11
18	22
19	8
20	7
21	2
22	0
23	13
24	9
25	12

Tabel 1. Nilai n dan Xn untuk a = 5, b = 13, M = 23 dan $X_0 = 0$

Dari tabel di atas, tampak bahwa barisan bilangan berulang lagi pada n = 22. Ini membuktikan bahwa bilangan acak semu memiliki suatu periode tertentu dan akan kembali ke keadaan awal begitu periode terlewati.

Keunggulan yang dimiliki LCG dibanding pembangkit bilangan semu lainnya adalah cepat dan hanya membutuhkan sedikit operasi bit. *LCG* banyak diterapkan untuk aplikasi simulasi sebab *LCG* mangkus dan memperlihatkan sifat statistik yang bagus dan sangat tepat untuk uji-uji empirik. Kekurangannya terletak pada ketidakcocokannya digunakan untuk kriptografi karena bilangan acak yang dihasilkan dapat dengan mudah diprediksi urutan kemunculannya.

Lebih jauh, walaupun LCG secara teoritis mampu menghasilkan bilangan acak yang tidak terlalu buruk, ia sangat sensitif terhadap pemilihan nilai-nilai a, b. dan m. Pemilihan nilai-nilai yang buruk dapat mengarah pada implementasi LCG yang tidak efisien. Tabel nilai konstanta yang bagus untuk LCG berdasarkan analisis dapat dilihat pada tabel 2.

a	b	m
106	1283	6075
211	1663	7875
421	1663	7875
430	2351	11979
936	1399	6655
1366	1283	6075
171	11213	53125
859	2531	11979
419	6173	29282
967	3041	14406
141	28411	134456
625	6571	31104
1541	2957	14000
1741	2731	12960
1291	4621	21870
205	29573	139968
421	17117	81000
1255	6173	29282
281	28411	134456

Tabel 2. Konstanta bagus untuk implementasi LCG

Sebenarnya, barisan bilangan ini juga dapat dibangkitkan dengan persamaan (2), yang merupakan bentuk eksplisit persamaan (1).

(2)

$$X_n = a^n X_0 + b\,(a^n - 1) / (a - 1) \pmod m$$

Persamaan (2) ini dapat kita buktikan dengan induksi matematika. Untuk k = 1 , persamaan tersebut jelas benar karena :

$$X_1 = aX_0 + b \pmod m$$

Asumsikan persamaan juga benar untuk nilai ke- k sehingga:

$$X_n = a^n X_0 + b\,(a^n - 1) / (a - 1) \pmod m$$

Karena

$$X_{k+1} = a\,X_n + b \pmod m$$

maka X_k dapat disubstitusi ke dalam persamaan:

$$X_{n+1} = a(a^n X_0 + b(a^n - 1)/(a-1)) + b$$
$$= a^{n+1} X_0 + b(a(a^n - 1)/(a-1) + 1)$$

$= a^{n+1} X_0 + b(a^{n+1} - a)/(a-1) + (a-1)/(a-1))$
$= a^{n+1} X_0 + b(a^{n+1} - 1)/(a-1) \pmod m$

Persamaan di atas sesuai dengan persamaan (2) sehingga telah terbukti bahwa persamaan (2) berlaku untuk semua bilangan bulat positif k.

4. BLUM BLUM SHUT

Blum Blum Shub (BBS) adalah sebuah *pseudorandom number generator* yang dibuat pada tahun 1986 oleh Lenore Blum, Manuel Blum, dan Michael Shub. BBS mengambil bentuk sebagai berikut:
(3)

$$X_{n+1} = (X_n)^2 \bmod M$$

di mana :

X_{n+1} = bilangan acak ke-n+1 dari deretnya
X_n = bilangan acak sebelumnya
M = modulus di mana *M = pq, yaitu* hasil kali dari dua bilangan prima besar *p* dan *q*.

Untuk membangkitkan bilangan acak dengan BBS, digunakan algoritma sebagai berikut:

1. Pilih dua buah bilangan prima rahasia p dan q yang masing-masing kongruen dengan 3 modulo 4 (hal ini akan menjamin tiap *quadratic residue* memiliki satu akar kuadrat yang juga merupakan *quadratic residue*) dan GCD(φ(*p*-1), φ(*q*-1)) harus kecil (hal ini mengakibatkan panjang siklus menjadi besar).
2. Kalikan kedua bilangan menjadi M = pq. Bilangan M ini disebut bilangan bulat Blum.
3. Pilih bilangan bulat acak lain, s, sebagai bibit sedemikian hingga:
   a. $2 \le S < n$
   b. S dan n relatif prima
   Kemudian hitung $X_0 = s^2 \bmod n$
4. Barisan bit acak dihasilkan dengan melakukan iterasi berikut sepanjang yang diinginkan:
   a. Hitung $X_{n+1} = X_n^2 \bmod M$
   b. Z_i = bit LSB (*Least Significant Bit*) dari X_i
   Barisan bit acak adalah $Z_1, Z_2, Z_3, \ldots$

Dalam tiap langkah dari algoritma ini, beberapa keluaran dihasilkan dari X_n. Keluaran tersebut secara umum adalah pasangan bit dari X_n atau satu atau lebih bit yang signifikan dari X_n. Sementara itu, simbol φ pada langkah 1 disebut sebagai tolient. Notasi φ(n) atau tolient dari sebuah bilangan bulat n didefinisikan sebagai jumlah bilangan bulat positif yang kurang dari atau sama dengan n dan coprime dengan n. Misalnya, φ(8) = 4 semenjak empat bilangan 1, 3, 5, dan 7 adalah coprime dari 8.

Satu hal menarik yang perlu diperhatikan dalam penggunaan PRNG Blum Blum Shut ini adalah bahwa kita tidak perlu melakukan iterasi untuk mendapatkan bilangan acak jika p dan q diketahui, sebab X_i dapat dihitung secara langsung menggunakan persamaan:
(4)

$$x_i = \left(x_0^{2^i \bmod (p-1)(q-1)}\right) \bmod M.$$

Untuk mengetahui cara pembangkitan bilangan acak semu dengan BBS dengan lebih baik, kita misalkan bilangan-bilangan yang dipilih adalah p = 11 dan q = 7 sehingga n = pq = 77. Sedangkan s yang dipilih adalah s = 3 dan kita hitung $X_0 = 3^2 \bmod 77 = 9$. Kemudian :

- $X_1 = X_0^2 \bmod n = 9^2 \bmod 77 = 4 \rightarrow Z_1 = 0$
- $X_2 = X_1^2 \bmod n = 4^2 \bmod 77 = 16 \rightarrow Z_2 = 0$
- $X_3 = X_2^2 \bmod n = 16^2 \bmod 77 = 25 \rightarrow Z_3 = 1$
- $X_4 = X_3^2 \bmod n = 25^2 \bmod 77 = 9 \rightarrow Z_4 = 1$
- $X_5 = X_4^2 \bmod n = 9^2 \bmod 77 = 4 \rightarrow Z_5 = 0$
- $X_6 = X_5^2 \bmod n = 4^2 \bmod 77 = 16 \rightarrow Z_6 = 0$
- $X_7 = X_6^2 \bmod n = 16^2 \bmod 77 = 25 \rightarrow Z_7 = 1$
- $X_8 = X_7^2 \bmod n = 25^2 \bmod 77 = 9 \rightarrow Z_8 = 1$

Terlihat bahwa terjadi pengulangan setiap empat kali iterasi sehingga barisan bit acak yang dihasilkan adalah 0011, 0011, 0011, …

Bilangan acak yang dihasilkan tidak harus 1 bit LSB tetapi bisa juga k-bit (k adalah bilangan bulat positif yang tidak melebihi $\log_2(\log_2 n)$). Misalkan kita memilih p = 11351 dan q = 11987 sehingga n = pq = 136064437. Lalu kita pilih s = 80331757 dan k = 4 (k tidak melebihi $\log_2(\log_2 136064437) = 4.75594$). Kemudian kita hitung:

$X_0 = 80331757^2 \bmod 136064437 = 1312737111$.

Barisan bit acak yang dihasilkan adalah sebagai berikut:

• $X_1 = X_0^2 \bmod n$
  $= 131273718^2 \bmod 136064437$
  $= 47497112$
  $Z_1 = 47497112 \equiv 8 \bmod 2^4$
  $= 1000_2$ (4 bit LSB dari 47497112)
• $X_2 = X_1^2 \bmod n$
  $= 47497112^2 \bmod 136064437$
  $= 69993144$
  $Z_2 = 69993144 \equiv 8 \bmod 2^4$
  $= 1000_2$ (4 bit LSB dari 69993144)
• $X_3 = X_2^2 \bmod n$
  $= 69993144^2 \bmod 136064437$
  $= 13801821$
  $Z_3 = 13801821 \equiv 5 \bmod 2^4$
  $= 0101_2$ (4 bit LSB dari 69993144)
…

Barisan bit acak yang dihasilkan:
1000, 1000, 0101 …

atau dalam basis 10:
8, 8, 5 …

Berbeda dengan LCG, generator ini justru tidak cocok digunakan untuk simulasi karena tidak begitu cepat. Namun demikian, BBS memiliki bukti keamanan yang baik. BBS mengaitkan kualitas generator dengan kerumitan komputasi atas sebuah pemfaktoran bilangan bulat. Ketika kedua bilangan prima dipilih dengan benar dan O (log log M) lower-order bits dari tiap Xn merupakan keluaran, maka dalam limit M yang bertambah besar, membedakan bit keluaran dari angka acak akan sesulit memfaktorisasikan M.

Jika pemfaktoran bilangan bulat sudah rumit, maka BBS dengan M yang besar akan memiliki sebuah keluaran yang bebas dari pola tak acak apapun yang dapat dicari dengan sejumlah perhitungan. Hal ini membuat BBS menjanjikan keandalan dalam teknologi kriptografi, terutama yang berkaitan dengan masalah faktorisasi, misalnya enkripsi RSA.

5. BEBERAPA GENERATOR LAIN

LCG dan BBS merupakan *pseudorandom number generator* dengan dasar teori bilangan bulat yang sudah sangat populer dan banyak diimplementasikan. Selain kedua generator ini, berikut contoh beberapa generator lain dengan dasar teori yang sama dan telah dipublikasikan:

- Quadratic Congruential Generator yang mengambil bentuk berikut :
  (5)
  $$X_{n+1} = (a X^2_n + b X_n + c) \bmod m$$

- Additive Number Generator yang dibuat oleh Mitchell and Moore pada tahun 1958 dan memiliki rumusan :
  (6)
  $$X_n = (X_{n-24} + X_{n-55}) \bmod m \quad , n \geq 5$$

  m genap, $X_0 \ldots X_{54}$ tidak semuanya genap. LSB dari "X_n mod 2" memiliki periode sebesar $2^{55} - 1$. Oleh karena itu, generator ini akan memiliki periode setidaknya sebesar ini.
- Inversive Congruential Generator yaitu generator yang menggunakan balikan modulo (jika ada) untuk untuk menghasilkan bilangan selanjutnya dalam barisan. Generator ini memiliki rumus standar:
  (7)
  $$X_{n+1} \equiv (aX_n^{-1} + c) \pmod m, \ 0 \leq X_n < m$$

6. KESIMPULAN

6.1 Tidak mungkin menghasilkan bilangan acak murni melalui proses komputasi. Bilangan acak yang dihasilkan dengan cara ini disebut bilangan acak semu.

6.2 Pembangkit bilangan acak semu yang menggunakan teori aritmatika modulo memiliki bentuk umum $X_{n+1} = f(X_n, X_{n-1}, \ldots, X_0) \bmod m$ di mana bilangan acak ke-(n+1) merupakan fungsi dari bilangan acak sebelumnya modulo m.

DAFTAR REFERENSI

[1]Munir, Rinaldi, Diktat Kuliah IF 2153, Matematika Diskrit, Edisi Keempat, Program Studi Teknik Informatika, STEI, ITB, 2006.

[2]Michael Luby, Pseudorandomness and Cryptographic Applications, Princeton Univ Press, 1996.

[3]http://en.wikipedia.org/wiki/Pseudorandom_number_generator. Tanggal akses 3 Januari 2009 pukul 10.00.

[4]http://en.wikipedia.org/wiki/Linear_congruential_generator. Tanggal akses 3 Januari 2009 pukul 10.00.

[5]http://en.wikipedia.org/wiki/Blum_Blum_Shub_generator. Tanggal akses 3 Januari 2009 pukul 10.00.