Analisis Keacakan Generator Angka Pseudorandom Mersenne Twister dengan Metode Diehard Test

Luqman A. Siswanto (13513024)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganesha 10-12, Bandung 40132, Indonesia
l.arifin.siswanto@gmail.com

Abstract **–** Mersenne Twister adalah algoritma untuk membangkitkan bilangan acak (PRNG/*Pseudo Random Number Generator*) yang popular digunakan karena penggunaan memorinya yang sangat kecil (2,5 KB). Salah satu implementasi Mersenne Twister yakni MT19937 memiliki periode yang sangat besar, $2^{19937} - 1$. Algoritma ini diklaim aman dan tidak bisa diprediksi walaupun sebenarnya periode besar tidak menjamin kualitas pembangkitan bilangan acak. Apakah klaim tersebut benar? Makalah ini akan mengulas tuntas kinerja Mersenne Twister menggunakan Diehard Test yang merupakan metode pembuktian secara statistik untuk mengukur kualitas generator bilangan acak.

Keywords **–** bilangan acak, *Diehard Test*, *Mersenne Twister*, RPNG

I. Pendahuluan

Pada masa ini teknologi komputer berkembang dengan sangat cepat. Apalagi pada era informasi ini, perkembangan teknologi komputer berkembang ke arah informasi. Analisis informasi memberikan banyak manfaat. Saat ini dikenal sebuah teknologi yang bernama Big Data. Dalam teknologi tersebut banyak informasi yang disimpan oleh sebuah instansi yang nantinya akan dianalisa menjadi pengetahuan yang sangat berharga. Pengetahuan tersebut akan membantu instansi tersebut meningkatkan performansinya. Masyarakat juga merasakan majunya teknologi dalam arah informasi. Dengan mudah masyarakat mendapatkan informasi akan sesuatu hal yang ingin dicari.

Penemuan komputer membawa manusia ke era yang benar-benar baru. Di abad 21 ini, dengan dikembangkannya komputer dan internet, peradaban manusia memasuki era baru, era digital. Banyak terjadi perubahan pola bisnis dan gaya hidup dengan beragam kemunculan teknologi baru di abad ini. Dengan adanya teknologi baru ini, manusia dihadapkan dengan berbagai kemudahan dan manfaat. Namun di sisi lain, kejahatan dan penipuan juga semakin mudah dilakukan.

Akan tetapi perkembangan teknologi ke arah informasi juga memberi dampak yang buruk. Banyak teknologi yang dengan mudah mengambil informasi yang bertebaran di jaringan internet. Data privasi seseorang bisa dengan mudah diambil oleh orang lain. Maka dari itu teknologi keamanan informasi juga harus dikembangkan sejajar dengan perkembangan teknologi informasi.

Keamanan digital kini menjadi sebuah kebutuhan esensial. Semenjak berubahnya era hidup manusia dari era modern ke era digital, keamanan digital menjadi sama pentingnya seperti keamanan konvensional. Banyak kebocoran informasi dari sebuah perusahaan yang diberitakan di publik yang menyebabkan kerugian besar. Oleh karena itu, setiap pribadi dan korporasi selayaknya memperhatikan keamanan digital dengan serius.

Kriptografi adalah salah satu solusi keamanan digital. Dengan kriptografi, dapat dimungkinkan penyembunyian informasi (*information hiding*) dan sertifikasi digital sehingga permasalahan autentikasi dan autorisasi dapat diselesaikan.

Beberapa elemen pada kriptografi membutuhkan generator untuk pembangkitan bilangan secara acak. Semakin acak bilangan dapat dihasilkan, maka bilangan yang muncul selanjutnya semakin tidak dapat diprediksi. Pada prinsip ketidak-dapat-diprediksian ini, kita akan memanfaatkannya sebagai landasan untuk keamanan penyembunyian informasi. Barisan bilangan random yang tidak dapat ditebak (random murni) akan memberikan keamanan terbaik.

Pada kenyataannya, tidak ada pembangkit bilangan random murni. Semua bilangan yang dibangkitkan dengan komputer hanya sekedar pseudo-random, artinya nanti pada suatu saat dengan periode tertentu akan berulang. Berbagai pendekatan dilakukan para *computer scientist* untuk mendapatkan hasil bilangan random yang berkualitas. Salah satu pembangkit bilangan acak yang populer adalah Mersenne Twister.

II. Dasar Teori

A. Pseudo-random Number Generator (PRNG)

Bilangan acak adalah bilangan yang kemunculan berikutnya tidak dapat diprediksi. Konsep bilangan acak ini

banyak digunakan dalam kriptografi, misalnya untuk pembangkitan parameter kunci pada algoritma kunci-public seperti Elliptic Curve dan RSA, atau dalam pembangkitan initialization vectorpada algoritma kunci-simeteri, dan masih banyak lainnya.

Pada dasarnya tidak ada komputasi yang benar-benar menghasilkan deret bilangan acak secara sempurna. Pembangkitan bilangan acak ini dapat didekati dengan bilangan acak semu, dengan batasan parameter tertentu yang didefinisikan sebagai 'syarat cukup' bilangan acak.

Bilangan acak yang dihasilkan dengan formula matematika disebut bilangan acak semu (pseudorandom number) karena pembangkitan bilangannya dapat diulang kembali. Pembangkit bilangan yang menghasilkan bilangan pseudorandom disebut pseudorandom number generator (PRNG).

Barisan yang dihasilkan dari PRNG sejatinya tidak benar-benar random karena sepenuhnya ditentukan dari sekumpulan nilai awal yang disebut *seed*. Meskipun begitu, barisan ini dapat didekati dengan seed berdasarkan waktu (one-time pad) atau seed yang dihasilkan dari keunikan hardware tertentu.

Lembaca Federal Keamanan Informasi Jerman (*Bundesamt für Sicherheit in der Informationstechnik/BSI*) menggagaskan empat kriteria kualitas pembangkit bilangan acak.

1. Berulangnya elemen identik berpeluang kecil.
2. Tidak dapat dibedakan dari bilangan random murni berdasarkan tes statistik tertentu.
3. Kemunculan bilangan random berikutnya/sebelumnya tidak dapat ditebak/dihitung.
4. Inner state dari pembangkit tidak dapat ditebak.

Contoh pembangkit bilangan pseudorandom acak yang paling sederhana adalah LCG (Linear Congruential Generator) dengan formula sebagai berikut.

$$X_n = (a_{X-1} + b) \bmod m$$

X_n = bilangan acak ke-n dari deretnya

X_{n-1} = bilangan acak sebelumnya

a = faktor pengali

b = increment

m = modulus

X_0 = seed

B. Mersenne Twister

Marsenne Twister adalah pembangkit bilangan pseudorandom yang paling luas digunakan untuk general-purpose. Nama Mersenne diambil dari panjang periode yang dipilih yang merupakan bilangan prima Mersenne.

MT dikembangkan pada tahun 1997 oleh Makoto Matsumoto dan Takuji Nishimura, dari Departemen Matematika, Keio University, Jepang. MT didesain secara khusus berdasarkan bug yang ditemukan di pembangkit bilangan pseudorandom sebelumnya. MT merupakan pembangkit pertama yang mampu membangkitkan bilangan pseudorandom berkualitas tinggi dengan cepat.

MT memiliki properti sebagai berikut:

- periode yang besar
- bilangan random terdistribusi dengan baik
- memori efisien
- cepat

Sebagai contoh, implementasi MT19937 yang biasa digunakan memiliki properti detil sebagai berikut.

- periode $2^{19937} - 1$
- 623 dimensi, rata terdistribusi dengan akurasi 32-bit
- hanya membutuhkan memori 624 buah 32-bit (2,5 KB)
- empat kali lebih cepat dari rand() di C

Berikut adalah parameter algoritma Mersenne Twister

w: ukuran bit word

n: derajat rekurensi

m: word tengah, $1 \le m < n$

r: titik pemisah dalam satu word, $0 \le r \le w - 1$

a: koefisien untuk twist matriks

b, c: konstanta bitmask

s, t: konstanta pergeseran bit

u, d, l: konstanta Mersenne Twister tambahan

Parameter untuk implementasi MT19937 adalah sebagai berikut.

w = 32, n = 624, m = 397, r = 31, a = $9908B0DF_{16}$, u = 11,

d = $FFFFFFFF_{16}$, s = 7, b = $9D2C5680_{16}$, t = 15,

c = $EFC6000016$, l = 18

Berikut ini adalah contoh implementasi Mersenne Twister dalam bahasa Go.

```
int[0..n-1] MT
int index := n+1
const int lower_mask = (1 << r) - 1
const int upper_mask = lowest w bits of
                       (not lower_mask)

function seed_mt(int seed) {
```

```
  index := n
  MT[0] := seed
  for i from 1 to (n - 1) {
    MT[i] := lowest w bits of (f * (MT[i-1]
             xor (MT[i-1] >> (w-2))) + i)
  }
}

function extract_number() {
  if index >= n {
    twist()
  }

  int y := MT[index]
  y := y xor ((y >> u) and d)
  y := y xor ((y << s) and b)
  y := y xor ((y << t) and c)
  y := y xor (y >> l)

  index := index + 1
  return lowest w bits of (y)
}

function twist() {
  for i from 0 to (n-1) {
    int x := (MT[i] and upper_mask)
           + (MT[(i+1) mod n] and lower_mask)
    int xA := x >> 1
    if (x mod 2) != 0 {
      xA := xA xor a
    }
    MT[i] := MT[(i + m) mod n] xor xA
  }
  index := 0
}
```

.

C. *Diehard Test*

Diehard test adalah pengujian secara statistik untuk mengukur kualitas yang dihasilkan oleh pembangkit bilangan pseudorandom. Diehard test dikembangkan oleh Professor George Marsaglia pada tahun 1995 dari Florida State University.

Detil langkah-langkah yang dilakukan dalam Diehard test dijabarkan di bab 3.

III. Metode Analisis

Kami memilih metode Diehard test karena Diehard dapat dibuktikan secara valid berdasarkan statistik. Diehard juga banyak digunakan sebagai metode untuk pengujian PRNG lain. Ide metode Diehard test terinspirasi dari metode lain yang lebih trivial.

Ada 2 metode yang akan diulas dalam makalah ini yakni metode Kendall and Smith Test dan Diehard Test. Kendall-Smith Test merupakan uji kualitas PRNG yang lebih sederhana, sementara Diehard melibatkan penghitungan dan formula statistik yang lebih *advanced*.

A. *Kendall and Smith Test*

Metode uji ini adalah metode yang paling trivial dalam pengujian kualitas bilangan acak. Metode ini adalah metode pertama kalinya yang pernah dipublikasi manusia dalam pengujian PRNG. Kendall and Smith Test dibuat menggunakan alat bantu statistik seperti chi-squared yang disempurnakan oleh Kendall-Smith.

Ide dasar Kendal-Smith Test bermula dari uji hipotesis bahwa setiap barisan bilangan random selalu memiliki frekuensi kemunculan yang merata, serta pola yang terjadi pada data seharusnya juga terdistribusi secara merata.

Kendall and Smith Test terdiri dari 4 uji:

1. Frequency test, merupakan test yang paling trivial. Menghitung kemunculan suatu bilangan dalam barisan.
2. Serial test, sama seperti frequency test tapi menghitung dua digit dalam waktu yang bersamaan. Semakin rata persebaran, maka semakin bagus.
3. Poker test, menguji barisan dalam 5 angka sekaligus, terinspirasi dari game poker.
4. Gap test, mencari jarak posisi kemunculan suatu bilangan, misalkan jarak 0 dalam barisan bilangan random hingga muncul bilangan 0 lagi. Jarak ini harus menghasilkan distribusi tertentu.

.

B. *Diehard Test*

Diehard Test adalah uji statistik untuk mengukur kualitas barisan bilangan pseudorandom yang dibangkitkan. Uji ini dikembangkan oleh George Marsaglia, peneliti dari Florida State University dan pertama kali dipublikasi pada 1995 dengan sebuah CD-ROM.

Berikut adalah sub-uji pada Diehard Test.

1. Birthday spacing. Ambil titik random pada interval yang panjang. Spasi antar pasang titik harus terdistribusi eksponensial asimpotik.
2. Overlapping permutations. Analisis setiap 5 bilangan random yang berurutan. 120 kemungkinan urutan seharusnya muncul dengan probabilitas merata.
3. Rank of matrices. Pilih beberapa bit dari beberapa angka random yang dibangkitkan untuk membentuk matriks (0, 1). Kemudian tentukan rank matriksnya. Hitung rank-nya.
4. Monkey test. Perlakukan bilangan random yang dibangkitkan sebagai “word”. “Word” yang tidak muncul akan mengikuti distribusi tertentu.
5. Count the 1s. Hitung bit 1 pada representasi bit angka yang berurutan. Konversikan hitungan ke barisan “karakter”. Hitung kemunculan setiap word yang berisi 5 “karakter” yang berurutan.

6. Parking lot test. Secara random taruh 1 unit lingkaran pada petak 100 x 100. Sebuah lingkaran dikatakan berhasil diparkirkan apabila tidak overlap dengan lingkaran yang berhasil diparkirkan sebelumnya. Setelah 12.000 kali percobaan, jumlah lingkaran yang berhasil diparkirkan seharusnya mengikuti distribusi normal.
7. Minimum distance test. Secara random letakkan 8.000 titik pada petak 10.000 x 10.000 kemudian cari jarak minimum antar titik. Kuadrat dari jarak ini seharusnya terdistribusi secara eksponensial.
8. Random spheres test. Secara random pilih 4.000 titik pada sebuah kubus dengan sisi 1.000. Cari sebuah titik pusat yang mana jarak ke seluruh titik harus minimum. Jarak minimum ini harus terdistribusi secara eksponensial.
9. The squeeze test. Kalikan 2^{31} dengan pecahan random antara (0, 1) berkali-kali hingga mencapai 1. Ulangi ini sebanyak 100.000 kali. Banyak pecahan yang dibutuhkan untuk mencapai seharusnya mengikuti distribusi tertentu.
10. Overlapping sums test. Bangkitkan sebuah barisan panjang berisi pecahan random antara (0, 1). Tambahkan barisan dengan 100 pecahan berurutan yang telah dibangkitkan sebelumnya. Jumlah barisan yang baru seharusnya terdistribusi secara normal.

IV. Eksperimen dan Hasil

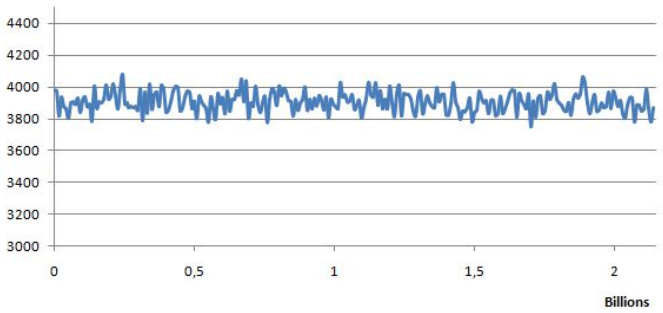
Untuk keperluan pengujian, kami menggunakan komputer dengan prosesor Intel i5-3317CPU @1.70GHz, RAM 8,00 GB, OS Windows 7-64 bit. Seluruh source code untuk keperluan pengujian ini dipublikasi secara open source dan dapat diakses di lokasi berikut:

https://github.com/luqmanarifin/mersenne-twister

Kami melakukan pengujian dengan Kendall-Smith test dan Diehard Test. Untuk Kendall-Smith, digunakan metode frequency test, serial test, poker test. Sementara Diehard Test menggunakan metode overlapping permutations, minimum distance set, parking lot test, overlapping sums test, dan the squeeze test.

A. Frequency Test

Dilakukan pembangkitan bilangan acak sebanyak 1.000.000 buah dengan range [0, 2147483647], kemudian dilihat persebarannya. PRNG yang bagus akan membangkitkan bilangan yang persebarannya merata. Berikut ini adalah hasilnya.

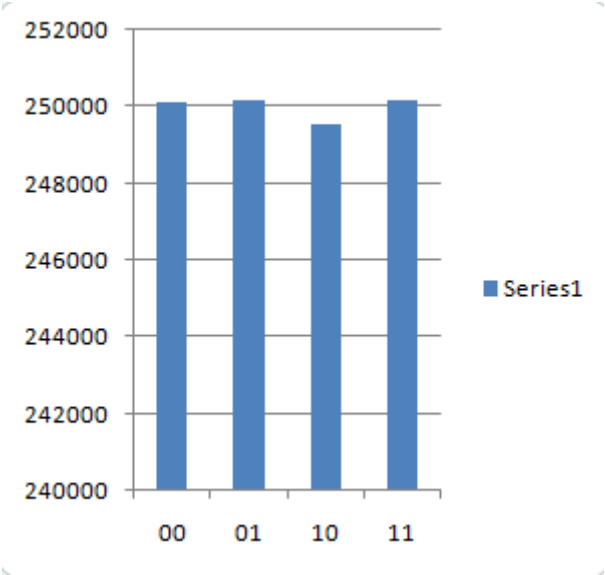


Gambar 1 - Grafik frekuensi kemunculan bilangan random dan persebarannya

Dapat dilihat bahwa persebaran bilangan terdistribusi merata ke semua interval.

B. Serial Test

Serial Test adalah pengujian persebaran frekuensi bit dari angka yang dibangkitkan. Serial test mirip dengan frekuensi test namun memperhatikan 2 bit yang berurutan. Terdapat 4 kemungkinan yakni 00, 01, 10, 11. Kami membangkitkan 1.000.000 buah bit dan melihat persebaran frekuensi serialnya.

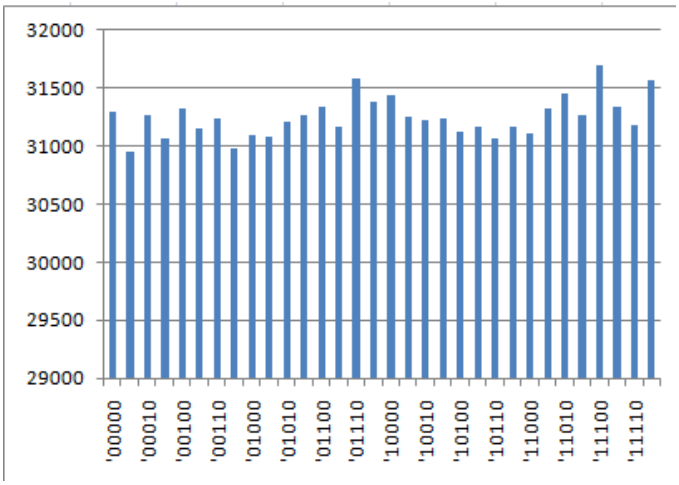


Gambar 2 – Grafik frekuensi serial bilangan acak

Dapat dilihat bahwa frekuensi serial bit yang bersebelahan tersebar secara merata ke semua kemungkinan.

C. *Poker Test*

Poker test adalah pengujian frekuensi bit dengan cara pengelompokan lima bit-lima bit. Uji ini mirip dengan uji poker test namun namun 5 bit. Ada 32 kemungkinan hasil yakni 00000, 00001, 00010, …, hingga 11111.

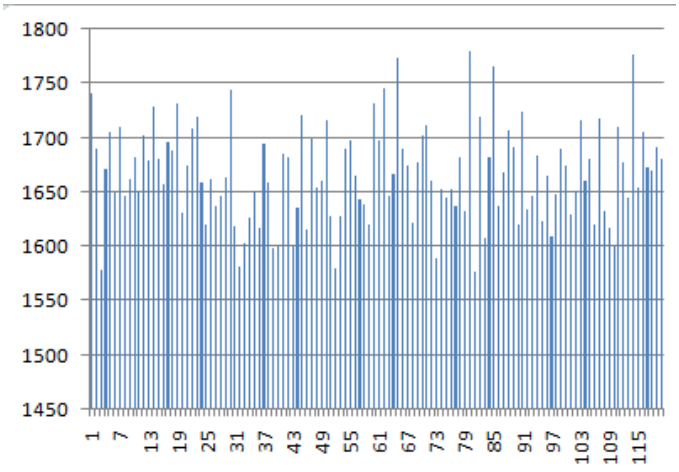


Gambar 3 - Grafik persebaran frekuensi kemunculan konfigurasi poker terhadap 32 kemungkinan poker hand yang mungkin

Untuk 1.000.000 kali percobaan, stream untuk setiap 5 bit (poker test) tersebar merata ke seluruh kemungkinan.

D. *Overlapping Permutations*

Overlapping permutations adalah teknik analisis untuk setiap 5 bilangan random yang berurutan. Terdapat 120 permutasi urutan perbandingan besar bilangan. PRNG yang baik akan menghasilkan seluruh kemungkinan secara merata. Uji ini dilakukan dengan cara membangkitkan 1.000.000 bilangan secara acak kemudian melakukan penghitungan frekuensi perbandingan setiap 5 bilangan yang berurutan.

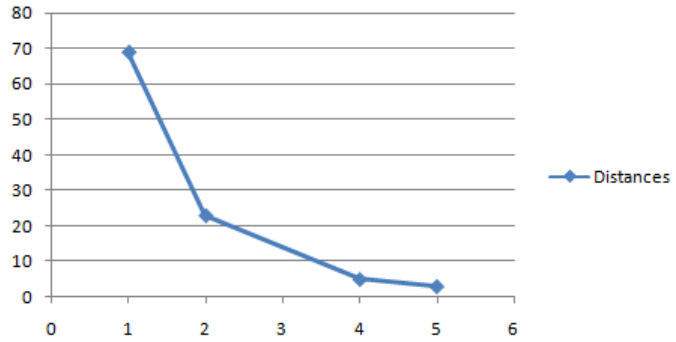


Gambar 4 – Grafik persebaran frekuensi kemunculan 120 permutasi dari 5 bilangan acak yang berurutan

Setelah pembangkitan 1.000.000 bilangan, persebaran permutasi perbandingan bilangan setiap 5 bilangan ternyata tersebar secara merata.

E. *Minimum Distance Set*

Secara random letakkan 8.000 titik dengan koordinat bilangan cacah pada petak 10.000 x 10.000 kemudian cari jarak minimum antar titik. Kuadrat dari jarak ini seharusnya terdistribusi secara eksponensial. Uji ini kami ulangi sebanyak 100 kali supaya mendapat polanya.



Gambar 5 – Grafik persebaran frekuensi minimum distance antar tiap pasang titik dari 8.000 titik random

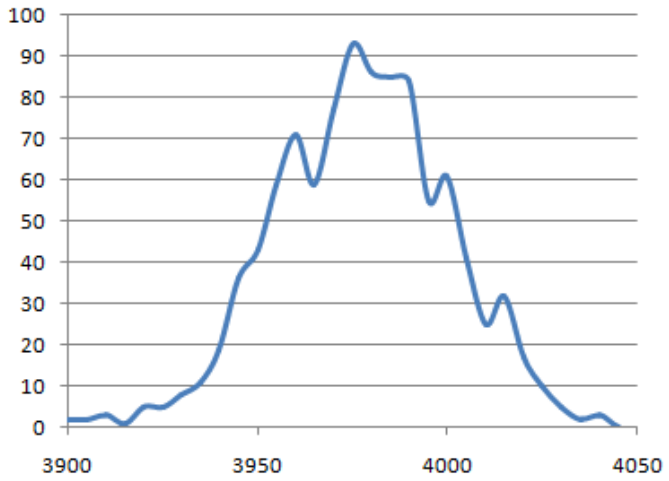
Graf tersebut menunjukkan bahwa jarak minimum tiap titik yang nilainya 1 muncul sebanyak 69 kali. Semakin besar nilai jarak minimum, maka kemunculannya pada set titik random juga semakin sedikit. Kemunculan jarak minimum antar pair titik terbukti terdistribusi secara eksponensial asimtotik sehingga MT dinyatakan lolos uji ini.

F. *Parking Lot Test*

Secara random taruh lingkaran berjari-jari 1 pada petak 100 x 100. Sebuah lingkaran dikatakan berhasil diparkirkan apabila tidak overlap dengan lingkaran yang berhasil diparkirkan sebelumnya. Apabila lingkaran tidak berhasil diparkirkan, abaikan. Apabila lingkaran berhasil diparkirkan, maka letakkan lingkaran pada tempat tersebut.

Setelah 12.000 kali penempatan lingkaran, hitung banyak lingkaran yang berhasil diparkirkan. Banyak lingkaran yang berhasil diparkirkan ini akan mengikuti distribusi normal.

Metode ini kami ulangi sebanyak 1.000 kali, kemudian berikut inilah hasil yang didapatkan.



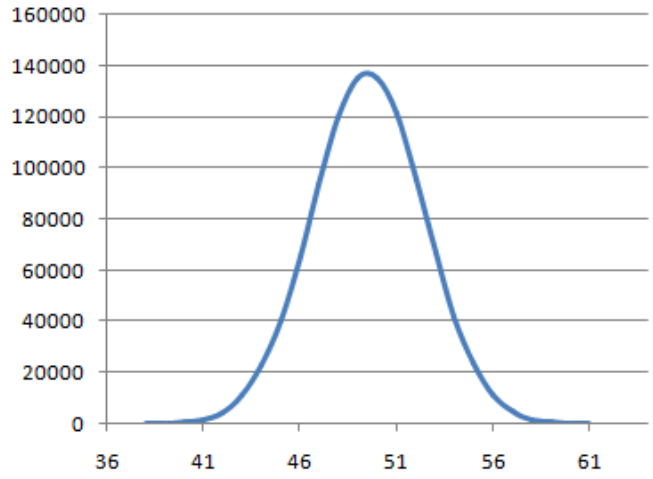
Gambar 6 – Grafik persebaran frekuensi banyak lingkaran yang berhasil parkir dalam uji Parking Lot test

Persebaran banyak lingkaran yang berhasil diparkirkan mendekati distribusi normal. Grafik yang dihasilkan tidak terlalu bagus tetapi hampir mendekati distribusi normal. Grafik di atas tidak mengikuti distribusi normal yang mulus karena dataset yang sedikit, yakni percobaan hanya dilakukan 1.000 kali tidak seperti uji lain yang dilakukan hingga mencapai 1.000.000 kali. Uji ini terbatas hingga 1.000 kali saja karena penulis belum menemukan algoritma yang lebih baik supaya komputasi lebih cepat.

G. Overlapping Sums

Bangkitkan sebuah barisan panjang berisi pecahan random antara [0, 1]. Jumlah dari 100 pecahan yang berurutan seharusnya mengikuti distribusi normal.

Kami melakukan pengujian dengan cara membangkitkan 1.000.000 buah pecahan dengan range [0, 1]. Kemudian jumlah dari 100 pecahan yang berurutan dihitung kemunculannya pada range untuk ditampilkan ke graf.



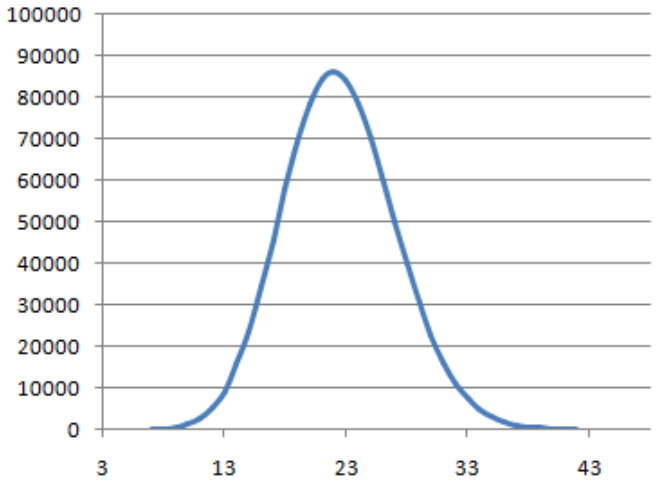
Gambar 7 – Grafik persebaran frekuensi jumlah dari 100 pecahan yang dibangkitkan berurutan

Persebaran jumlah 100 bilangan pecahan yang bersebalahan mengikuti distribusi normal sehingga MT dikatakan lolos uji ini.

H. Squeeze Test

Kalikan 2^{31} dengan pecahan random antara (0, 1) berkali-kali hingga mencapai 1. Ulangi ini sebanyak 100.000 kali. Banyak pecahan yang dibutuhkan untuk mencapai 1 seharusnya mengikuti distribusi normal.

Kami melakukan pengujian dengan cara mensimulasikan prosedur di atas sebanyak 1.000.000 kali.



Gambar 8 – Grafik persebaran frekuensi banyak bilangan pecahan yang dibutuhkan untuk mengalikan bilangan 2^{31} menjadi 1

Rata-rata jumlah bilangan pecahan yang dibutuhkan untuk mengalikan bilangan supaya mencapai 2^{31} adalah 23 kali. Grafik di atas mengikuti distribusi normal sehingga MT dapat dikatakan lolos uji Squeeze Test ini.

V. KESIMPULAN DAN SARAN

Algoritma Mersenne Twister memiliki performa yang baik, selain ringan karena hanya membutuhkan 2,5 KB memori, algoritma ini menghasilkan bilangan random berkualitas dan memiliki periode yang sangat panjang yakni $2^{19937} - 1$.

Menurut uji Diehard Test dan Kendall Smith Test, algoritma Mersenne Twister dapat dikategorikan sebagai PRNG berkualitas tinggi.

KATA PENUTUP

Makalah ini disusun untuk keperluan memenuhi tugas mata kuliah IF4020 Kriptografi di Institut Teknologi Bandung semester genap tahun ajaran 2015/2016.

Penulis ingin menghaturkan banyak terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. yang telah mengampu mata kuliah IF4020 Kriptografi selama satu semester ini.

REFERENSI

[1] M. Matsumoto and T. Nishimura, “Mersenne twister: a 623-dimensonally equidistributed uniform pseudo-random number generator”, in ACM Transaction on Modelling and Computer Simulation (TOMACS), vol. 8, Jan 1998, pp.3-30.

[2] M. Matsumoto and T. Nishimura, “Dynamic Creation of Pseudorandom Number Generators”, Department of Mathematics, Keio University, Japan.

[3] Walpole, et al, “Probability and Statistics for Engineers and Scientists 9th ed.”, Boston: Pearson Education, 2012.

[4] G. Marsaglia, “The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness”, Department of Statistics and Supercomputer Computations Research Institute, Florida State University.

[5] M. Hazewinkel, “Exponential Distribution”, Encyclopedia of Mathematics, Springer, ISBN 978-1-55608-010-4.

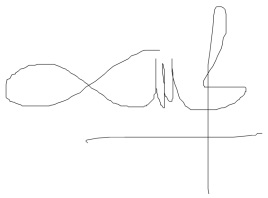
[6] M. Luby and C. Rackoff. "How to Construct Pseudorandom Permutations and Pseudorandom Functions." In SIAM J. Comput., vol. 17, 1988, pp. 373-386.

[7] R. Munir. “Pembangkit Bilangan Acak”, bahan kuliah IF4020 Kriptografi, Teknik Informatika, Institut Teknologi Bandung, 2015.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2016



Luqman A. Siswanto