Analisis Pengaruh Penggunaan Random Seed pada Linear Congruential Generator (LCG)

Nathan Tenka - 13521172[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13521172@itb.ac.id

***Abstract*—Pembangkit bilangan acak semu (PBAS) adalah salah satu algoritma yang banyak diperlukan dalam dunia teknologi informasi. *Video game*, algoritma acak (seperti algoritma Monte Carlo dan Las Vegas), kriptografi, dan berbagai bidang lain memerlukan bilangan acak. Untuk keperluan tertentu, diperlukan PBAS yang hasilnya mendekati bilangan acak yang sebenarnya. Salah satu cara untuk meningkatkan kualitas dan keamanan PBAS adalah menggunakan sumber acak sebagai *seed* pada PBAS. Makalah ini akan mengulas hasil dari penggunaan beberapa sumber sebagai *seed* pada PBAS *Linear Congruential Generator* (LCG).**

***Keywords*—PBAS, LCG, random seed, reseed**

I. Pendahuluan

Dalam dunia teknologi informasi seringkali diperlukan bilangan acak untuk berbagai keperluan. Misalnya pada *video game*, bilangan acak bisa digunakan untuk menentukan perilaku musuh supaya tidak monoton. Pada kriptografi, bilangan acak diperlukan untuk menentukan *key* yang akan digunakan untuk enkripsi maupun dekripsi. Masih banyak bidang lain yang memerlukan bilangan acak untuk menyelesaikan persoalan.

Akan tetapi, menghasilkan bilangan yang benar-benar acak pada komputer sangat sulit. Hal ini disebabkan komputer hanya mengikuti instruksi yang diberikan oleh program. Oleh karena itu, biasanya digunakan pembangkit bilangan acak semu (PBAS) yang menghasilkan angka-angka yang terlihat acak, walau sebenarnya angka tersebut memiliki pola tertentu. Untuk keperluan seperti kriptografi, diperlukan PBAS yang hasilnya hampir mustahil ditebak atau mendekati bilangan acak sesungguhnya. Salah satu cara untuk meningkatkan keamanan hasil PBAS adalah menggunakan sumber yang memang acak (seperti cuaca, bunyi atmosfer, dll) sebagai *seed* yang digunakan pada PBAS. Satu *Seed* dipakai untuk menghasilkan beberapa angka, lalu akan dilakukan *reseed* setelah interval tertentu untuk menjaga keamanan. Berikut akan diulas hasil dari penggunaan beberapa sumber *seed* pada PBAS berbasis LCG.

II. Dasar Teori

2.1. Teori Bilangan

Teori bilangan adalah cabang matematika murni yang mempelajari bilangan bulat [1]. Teori bilangan mencakup di antaranya [1]-[3]:

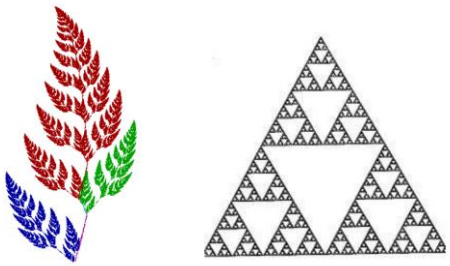
- Sifat pembagian pada bilangan bulat
- Teorema dan Algoritma Euclidean (membahas tentang pembagian dan algoritma untuk mendapatkan PBB dari dua bilangan bulat)
- Pembagi Bersama Terbesar (PBB)
- Aritmetika Modulo
- Sistem Kekongruenan Linier
- Bilangan Prima

Linear congruential generator (LCG) berhubungan dengan aritmetika modulo. Untuk suatu bilangan a dan m (m > 0), a mod m akan menghasilkan sisa dari pembagian a dengan m [1]. Hasil yang didapat pun berada dalam rentang [0..m-1]. Selain itu, dua bilangan disebut kongruen dalam modulo m jika keduanya memiliki hasil yang sama jika dimod dengan m [1].

Dalam artimetika modulo masih terdapat konsep-konsep lain seperti balikan modulo, namun konsep-konsep tersebut tidak diperlukan dalam LCG sehingga tidak akan dibahas lebih lanjut.

2.2. Relasi Rekurens

Rekursi adalah proses mendefinisikan suatu objek dalam arti dirinya sendiri [4]. Contoh rekursivitas terdapat dalam bentuk-bentuk fraktal yang ditunjukkan pada gambar di bawah ini.



Gambar 2.2.1 Contoh bentuk fraktal
Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Rekursi-dan-relasi-rekurens-(Bagian-1).pdf

Dalam dunia pemrograman, rekursi biasa digunakan dalam membuat fungsi rekursif, yaitu fungsi yang memiliki dua bagian [4] :

a. Basis, bagian fungsi yang mengandung nilai terdefinisi sekaligus berguna untuk menghentikan rekursi.
b. Rekurens, bagian yang mendefinisikan fungsi berdasarkan dirinya sendiri. Rekurens menggunakan parameter yang semakin lama semakin mendekati basis.

Relasi rekurens sendiri adalah persamaan yang menyatakan suatu barisan sebagai fungsi dari suku-suku sebelumnya. Relasi rekurens juga memiliki basis (disebut kondisi awal) yang akan menentukan isi barisan. LCG adalah salah satu contoh relasi rekurens.

2.3. *Linear Congruential Generator* (LCG)

LCG adalah salah satu algoritma PBAS tertua dan paling umum yang menghasilkan bilangan acak yang cukup baik. LCG memiliki bentuk persamaan berikut :

$$X_n = (aX_{n-1} + b) \bmod m$$

X_n = bilangan acak ke-n dari deretnya
X_{n-1} = bilangan acak sebelumnya
a = faktor pengali
b = *increment*
m = modulus
Kunci pembangkit adalah X_0 yang disebut **umpan** (*seed*).

Gambar 2.3.1 Persamaan *Linear Congruential Generator* (LCG)
Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian3.pdf

Terlihat bahwa LCG menghasilkan bilangan acak berdasarkan nilai yang didapat sebelumnya pada relasi rekurens sehingga bilangan yang dihasilkan dipengaruhi oleh pemilihan *seed* (X_0). Algoritma ini terkenal karena efisien dan mudah diimplementasikan. Walaupun begitu, masih terdapat banyak kekurangan yang menyebabkan LCG tidak cocok digunakan untuk bidang yang memerlukan keamanan tingkat tinggi. Contoh kelemahannya adalah nilai yang dihasilkan akan berulang setiap m bilangan / periodik (walau periodisitas berguna untuk kasus-kasus tertentu), sifat statistik yang kurang bagus, dan hasilnya sensitif terhadap nilai a dan m. Sebenarnya kelemahan-kelemahan tersebut bisa diatasi dengan pemilihan nilai a dan m yang tepat serta m yang cukup besar. Dengan pemilihan nilai yang tepat, LCG juga bisa mengalahkan hasil dari PBAS yang lebih rumit. Akan tetapi, LCG tetap kurang aman digunakan dalam bidang-bidang seperti kriptografi karena nilai a, b, dan m bisa ditebak dari bilangan yang dihasilkan. Oleh karena itu, muncul algoritma-algoritma PBAS lain yang berusaha mengatasi kekurangan tersebut.

2.4. *Random Seed*

Seperti yang telah disebutkan sebelumnya, salah satu cara untuk meningkatkan kualitas dan keamanan bilangan acak yang dihasilkan PBAS adalah dengan menggunakan sumber acak sebagai *seed* [8]. Sumber-sumber ini juga disebut sebagai sumber entropi. Semakin acak suatu sumber, semakin besar entropinya. Beberapa sumber entropi yang bisa digunakan adalah :

- Kamera dan mikrofon pada perangkat
- *Rotational Latency* dari *disk*
- *System clock*
- Waktu dan nilai dari suatu aksi eksternal
- Bunyi atmosferik

Sumber yang akan digunakan adalah kamera dan mikrofon serta *system clock* (memakai besaran jam, menit, detik, mikrodetik). Masing-masing sumber tersebut memiliki karakteristik tersendiri, di antaranya :

- *System clock*
  - Sangat bergantung pada perangkat keras dan sistem operasi.
  - Tidak bisa digunakan untuk melakukan *reseed* dalam interval yang terlalu cepat, tergantung pada ketelitian pengukuran waktu dari perangkat.
  - Memiliki jumlah bilangan acak yang relatif sedikit.
- Kamera dan mikrofon
  - Bisa memiliki entropi yang baik terutama karena hasil pembacaan dari lingkungan biasanya berbeda-beda.
  - Bergantung pada perangkat keras.
  - Waktu eksekusi lebih lama dibanding *system clock* karena harus mengakses data dari kamera atau mikrofon lebih dulu.

2.5. Pengujian Pembangkit Bilangan Acak Semu

Kualitas sebuah PBAS bisa diuji dengan beberapa skema pengujian. Salah satu cara adalah dengan melihat sifat statistik dari barisan yang dihasilkan PBAS. Beberapa uji statistik yang ada adalah :

- NIST Test Suite for Random Number Generators
- Diehard Test
- TestUO1
- PractRands

Uji-uji di atas adalah beberapa uji bilangan acack yang paling terkenal. Pada makalah ini hanya akan digunakan uji PBAS dari NIST (*National Institute of Standards and Technology*), secara spesifik versi SP800-22r1a [6] yang diimplementasikan oleh David Johnston [5]. Uji ini terdiri dari :

1. Frequency (Monobit) Test
   Menentukan kemiripan proporsi kemunculan bit 0 dan 1. Menjadi basis untuk lolos dari uji-uji yang lain.
2. Frequency Test Within A Block
   Menguji proporsi kemunculan 1 dalam blok-blok berukuran M bit. PBAS lolos jika proporsi ini mendekati M/2.
3. Runs Test
   Menguji banyak barisan berulang (*run*) dengan panjang beragam dalam barisan yang dimasukkan.
4. Test for the Longest Run of Ones in a Block
   Menguji panjang *run* bit 1 maksimum dalam blok berukuran M bit.
5. Binary Matrix Rank Test
   Menguji adanya ketergantungan linier antara substring panjang tertentu pada barisan masukan. Uji ini menggunakan rank submatriks *disjoint* dari barisan awal.
6. Discrete Fourier Transform (Spectral) Test
   Menguji sifat periodik barisan memakai puncak dari

Discrete Fourier Transform.

7. Non-overlapping Template Matching Test
   Menguji kemunculan string tertentu dalam barisan. Berfungsi untuk mendeteksi pola non-periodik yang muncul terlalu banyak dari hasil PBAS. Jika pola ditemukan, pencarian dilanjutkan dari setelah pola.
8. Overlapping Template Matching Test
   Mirip dengan Non-overlapping Template Matching Test tapi setiap kali pola ditemukan, pencarian dilanjutkan dari bit berikutnya, bukan setelah pola.
9. Maurer's "Universal Statistical" Test
   Menguji kompresibilitas barisan (barisan kompresibel bisa dipersingkat tanpa kehilangan informasi). Barisan yang sangat kompresibel dianggap tidak acak. Menggunakan banyak bit di antara pola.
10. Linear Complexity Test
    Menguji kompleksitas barisan berdasarkan panjang LFSR (*linear feedback shift register*). LFSR yang panjang menandakan bahwa barisan acak.
11. Serial Test
    Menguji frekuensi semua pola m bit yang saling tumpang tindih dalam barisan.
12. Approximate Entropy Test
    Mirip dengan Serial Test, tetapi untuk blok tumpang tindih yang panjangnya berbeda satu.
13. Cumulative Sums (Cusum) Test
    Menguji jumlah kumulatif dari bagian-bagian barisan (bisa dianggap sebagai *random walk*). Ekskursi dari *random walk* ini pada barisan acak akan mendekati nol.
14. Random Excursions Test
    Menguji jumlah *visit* pada suatu *state* dalam siklus *cumulative sum random walk*.
15. Random Excursions Variant Test
    Menguji jumlah *visit* pada *state-state* beragam dalam *cumulative sum random walk*.

Panjang barisan minimum yang diperlukan untuk masing-masing uji ada di rentang 100 – 1.000.000 bit. Untuk pengujian akan digunakan barisan dengan 1.000.000 bilangan bulat karena berdasarkan uji coba hasilnya lebih konsisten dibanding banyak bilangan yang lebih sedikit.

III. IMPLEMENTASI

Program menggunakan modul datetime, time, pyaudio, wave, opencv, dan sys. Pengolahan data untuk *seed* dari tiap sumber dilakukan dengan menjumlahkan tiap komponen hasil pembacaan yang kemudian dimodulo dengan 256 supaya didapat bilangan yang bisa direpresentasikan dalam 1 byte saja. Setiap hasil LCG juga dibuat berada dalam rentang [0..255]. Hal ini diperlukan karena angka hasil LCG perlu diubah ke dalam format binary. Untuk LCG dengan *reseed*, *reseed* dilakukan setiap didapat 200.000 (modulo m yang diambil) bilangan karena LCG memiliki periode sebesar modulo yang dipilih Hal ini menyebabkan akan terjadi lima kali *reseed* pada LCG karena diambil 1.000.000 bilangan. Nilai a dan b dipilih secara acak.

```
def timeSeed() :
    currDate = datetime.datetime.now()
    Hour = currDate.hour
    Minute = currDate.minute
    Second = currDate.second
    Micro = currDate.microsecond
    Total = Hour+Minute+Second+Micro
    print("Seed waktu :", Total % 256)
    return int(Total % 256)
```

Gambar 3.1 Fungsi generator *seed* dari waktu sistem
Sumber : Dokumen Penulis

```
def camSeed() :
    cam = cv2.VideoCapture(0)
    retCam, frame = cam.read()
    if retCam :
        cv2.imwrite('captured.jpg',frame)
    cam.release()
    camMtrx = cv2.imread('captured.jpg')
    sum = 0
    for Pixels in camMtrx :
        for Colors in Pixels :
            for values in Colors :
                sum += values
    print ("Seed kamera :",sum % 256)
    return int(sum % 256)
```

Gambar 3.2 Fungsi generator *seed* dari kamera
Sumber : Dokumen Penulis

```python
def audioSeed() :
    # Sumber : http://people.csail.mit.edu/hubert/pyaudio/
    CHUNK = 1024
    FORMAT = pyaudio.paInt16
    CHANNELS = 2
    RATE = 44100
    RECORD_SECONDS = 2
    WAVE_OUTPUT_FILENAME = "record.wav"

    p = pyaudio.PyAudio()

    stream = p.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    input=True,
                    frames_per_buffer=CHUNK)

    print("* recording")

    frames = []

    for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
        data = stream.read(CHUNK)
        frames.append(data)
    print("* done recording")
    sum = 0
    for values in frames :
        sum += int.from_bytes(values,sys.byteorder,signed=True)

    stream.stop_stream()
    stream.close()
    p.terminate()

    wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
    wf.setnchannels(CHANNELS)
    wf.setsampwidth(p.get_sample_size(FORMAT))
    wf.setframerate(RATE)
    wf.writeframes(b''.join(frames))
    wf.close()

    print("Seed suara :", sum % 256)
    return int(sum % 256)
```

Gambar 3.3.1 dan **3.3.2** Fungsi generator *seed* dari suara
Sumber : Dokumen Penulis

```python
def lcgNoReseed(n, a=4, b=3, m=200000) :
    start = time.time()
    if (n > 0) :
        totalArrRand = [timeSeed()]
        newNum = totalArrRand[0]
        for i in range(1,n) :
            newNum = ((a*newNum+b) % m)
            totalArrRand.append(int((256 * newNum)//m)) # an
        end = time.time()
        print("Execution time:", end-start)
        return totalArrRand
    else :
        return timeSeed()
```

Gambar 3.4 Fungsi LCG tanpa *reseed*
Sumber : Dokumen Penulis

```python
def lcgReseed(n, a=4, b=3, m=200000) : # Nilai a dan b dipilih sembarang
    # Random number generator berupa linear congruential generator
    start = time.time()
    if (n > 0) :
        totalArrRand = []
        currArrRand = [audioSeed()] # Sumber seed diganti sesuai sumber yang diuji
        newNum = currArrRand[0]
        currStop = 0
        for i in range(1,n) :
            if (i+1-currStop > m) : # Jika sudah mencapai modulo, lakukan reseed su
                for num in currArrRand :
                    totalArrRand.append(num)
                currArrRand = [audioSeed()]
                newNum = currArrRand[0]
                currStop = i
            newNum = ((a*newNum+b) % m)
            currArrRand.append(int((256 * newNum)//m)) # angka acak harus di antara
        for num in currArrRand :
            totalArrRand.append(num)
        end = time.time()
        print("Execution time:", end-start)
        return totalArrRand
    else :
        return audioSeed()
```

Gambar 3.5 Fungsi LCG dengan *reseed*
Sumber : Dokumen Penulis

```python
sequence = bytearray(lcgReseed(1000000))
f = open('rng_data.bin', 'wb')
f.write(sequence)
f.close()
```

Gambar 3.6 Program utama untuk menuliskan hasil LCG ke file *binary*
Sumber : Dokumen Penulis

IV. HASIL DAN PEMBAHASAN

4.1. Hasil uji LCG tanpa melakukan *reseed*

```
SUMMARY
-------
monobit_test                            0.5701661991269507 PASS
frequency_within_block_test             -7.48102325196193e-48 FAIL
runs_test                               0.9999090513659471 PASS
longest_run_ones_in_a_block_test        2.1338534155678214e-203 FAIL
binary_matrix_rank_test                 0.0                FAIL
dft_test                                0.0                FAIL
non_overlapping_template_matching_test  -1.730765619290071e-15 FAIL
overlapping_template_matching_test      0.0                FAIL
maurers_universal_test                  0.0                FAIL
linear_complexity_test                  5.153434406888992e-60 FAIL
serial_test                             0.22052296695471407 PASS
approximate_entropy_test                0.4680816591965474 PASS
cumulative_sums_test                    0.9365670563827859 PASS
random_excursion_test                   3.226616557026312e-44 FAIL
random_excursion_variant_test           0.28023863124670917 PASS
```

Gambar 4.1.1 Hasil uji pertama LCG tanpa melakukan *reseed* dengan *seed* 153
Sumber : Dokumen Penulis

```
SUMMARY
-------
monobit_test                          0.5672885173831944 PASS
frequency_within_block_test           -5.532360416149 39e-47 FAIL
runs_test                             0.9967071845516586 PASS
longest_run_ones_in_a_block_test      2.1338534155678214e-203 FAIL
binary_matrix_rank_test               0.0                FAIL
dft_test                              0.0                FAIL
non_overlapping_template_matching_test -6.485751999873491e-13 FAIL
overlapping_template_matching_test    0.0                FAIL
maurers_universal_test                0.0                FAIL
linear_complexity_test                4.2921151047699877e-60 FAIL
serial_test                           0.21974407393231582 PASS
approximate_entropy_test              0.46612717440275825 PASS
cumulative_sums_test                  0.9462295946773274 PASS
random_excursion_test                 3.9258479092684467e-32 FAIL
random_excursion_variant_test         0.7409138900854139 PASS
```

Gambar 4.1.2 Hasil uji kedua LCG tanpa melakukan *reseed* dengan *seed* 8
Sumber : Dokumen Penulis

```
SUMMARY
-------
monobit_test                          0.5740138986415111 PASS
frequency_within_block_test           2.1273139976485753e-48 FAIL
runs_test                             0.9982182871793845 PASS
longest_run_ones_in_a_block_test      2.1338534155678214e-203 FAIL
binary_matrix_rank_test               0.0                FAIL
dft_test                              0.0                FAIL
non_overlapping_template_matching_test 0.8818098271869815 PASS
overlapping_template_matching_test    0.0                FAIL
maurers_universal_test                0.0                FAIL
linear_complexity_test                3.986702367967466e-45 FAIL
serial_test                           5.139288170990959e-10 FAIL
approximate_entropy_test              1.4011968168156812e-09 FAIL
cumulative_sums_test                  0.9290157213932542 PASS
random_excursion_test                 0.0025321004989150 77 FAIL
random_excursion_variant_test         0.06924887875398542 PASS
```

Gambar 4.1.3 Hasil uji ketiga LCG tanpa melakukan *reseed* dengan *seed* 111
Sumber : Dokumen Penulis

Dari hasil di atas terlihat bahwa sifat statistik dari barisan bilangan yang dihasilkan LCG dengan nilai a, b, dan m yang digunakan relatif kurang baik. LCG berhasil melewati rata-rata 17/3 uji, kurang dari setengah jenis uji yang ada. Jenis uji yang berhasil dilewati juga dipengaruhi *seed* (terlihat dari perbedaan hasil antara *seed* 153 dan 8 dengan 111).

4.2. LCG dengan *reseed* dari waktu (Jam, menit, detik, mikrodetik)

```
Seed waktu : 196
Seed waktu : 52
Seed waktu : 143
Seed waktu : 109
Seed waktu : 132
Execution time: 0.3519754409790039
```

```
SUMMARY
-------
monobit_test                          0.6386994477342989 PASS
frequency_within_block_test           -1.4423282183243659e-43 FAIL
runs_test                             0.2614539337821151 PASS
longest_run_ones_in_a_block_test      2.1338534155678214e-203 FAIL
binary_matrix_rank_test               0.0                FAIL
dft_test                              0.0                FAIL
non_overlapping_template_matching_test 0.6876306390118985 PASS
overlapping_template_matching_test    0.0                FAIL
maurers_universal_test                0.0                FAIL
linear_complexity_test                0.0003698890360951784 FAIL
serial_test                           1.531514257640808e-07 FAIL
approximate_entropy_test              1.035872061640226e-06 FAIL
cumulative_sums_test                  0.989587162916135  PASS
random_excursion_test                 1.2531363273569787e-05 FAIL
random_excursion_variant_test         0.12557043909448726 PASS
```

Gambar 4.2.1 dan **4.2.2** *Seed* dan hasil uji pertama LCG dengan *reseed* menggunakan waktu
Sumber : Dokumen Penulis

```
Seed waktu : 125
Seed waktu : 226
Seed waktu : 32
Seed waktu : 73
Seed waktu : 40
Execution time: 0.3679845333099365
```

```
SUMMARY
-------
monobit_test                          0.5606018980085992 PASS
frequency_within_block_test           1.071093751640247e-41 FAIL
runs_test                             0.9859915427038065 PASS
longest_run_ones_in_a_block_test      2.1338534155678214e-203 FAIL
binary_matrix_rank_test               0.0                FAIL
dft_test                              0.0                FAIL
non_overlapping_template_matching_test 0.9988269178620571 PASS
overlapping_template_matching_test    0.0                FAIL
maurers_universal_test                0.0                FAIL
linear_complexity_test                3.9073428928879686e-11 FAIL
serial_test                           0.5573703968542246 PASS
approximate_entropy_test              0.5574795327319401 PASS
cumulative_sums_test                  0.9215234980514868 PASS
random_excursion_test                 2.956415795972443e-60 FAIL
random_excursion_variant_test         0.3761975434976571 PASS
```

Gambar 4.2.3 dan **4.2.4** *Seed* dan hasil uji kedua LCG dengan *reseed* menggunakan waktu
Sumber : Dokumen Penulis

```
Seed waktu : 202
Seed waktu : 230
Seed waktu : 254
Seed waktu : 23
Seed waktu : 85
Execution time: 0.3160746097564697
```

```
SUMMARY
-------
monobit_test                                0.6483303218539302 PASS
frequency_within_block_test                 3.3349246578228836e-46 FAIL
runs_test                                   0.25261411089990043 PASS
longest_run_ones_in_a_block_test            2.1338534155678214e-203 FAIL
binary_matrix_rank_test                     0.0                FAIL
dft_test                                    0.0                FAIL
non_overlapping_template_matching_test      0.9049145537558075 PASS
overlapping_template_matching_test          0.0                FAIL
maurers_universal_test                      0.0                FAIL
linear_complexity_test                      5.246181442777597e-06 FAIL
serial_test                                 0.1378111168940691 PASS
approximate_entropy_test                    0.2496483160601153 PASS
cumulative_sums_test                        0.9936752264635844 PASS
random_excursion_test                       9.124909181615444e-10 FAIL
random_excursion_variant_test               0.03520412428330144 PASS
```

Gambar 4.2.5 dan **4.2.6** *Seed* dan hasil uji ketiga LCG dengan *reseed* menggunakan waktu
Sumber : Dokumen Penulis

Terlihat bahwa melakukan *reseed* dengan waktu memiliki hasil yang lebih baik dibanding LCG tanpa melakukan *reseed*. Dengan *reseed* waktu, LCG berhasil melewati rata-rata 19/3 jenis uji, walau masih ada variasi pada salah satu hasil uji. Jenis uji yang konsisten berhasil dilewati oleh LCG dengan *reseed* tapi gagal di dua hasil uji LCG tanpa *reseed* adalah non-overlapping template matching test, yang menunjukkan bahwa pada LCG dengan *reseed* kemunculan pola tertentu dalam barisan mirip dengan barisan acak sesungguhnya. Hasil ini bisa dijelaskan karena tanpa *reseed*, setiap m bilangan akan terjadi pengulangan pola barisan, sedangkan dengan *reseed* pengulangan tersebut bisa dihindari.

4.3. LCG dengan *reseed* dari kamera

```
Seed kamera : 248
Seed kamera : 137
Seed kamera : 183
Seed kamera : 32
Seed kamera : 93
Execution time: 11.914024353027344
```

```
SUMMARY
-------
monobit_test                                0.5639408204652465 PASS
frequency_within_block_test                 -2.66783396862107e-44 FAIL
runs_test                                   0.9888105139218797 PASS
longest_run_ones_in_a_block_test            2.1338534155678214e-203 FAIL
binary_matrix_rank_test                     0.0                FAIL
dft_test                                    0.0                FAIL
non_overlapping_template_matching_test      1.4551054741230108e-10 FAIL
overlapping_template_matching_test          0.0                FAIL
maurers_universal_test                      0.0                FAIL
linear_complexity_test                      1.683647871762527e-05 FAIL
serial_test                                 0.5597413219126671 PASS
approximate_entropy_test                    0.5598510847781637 PASS
cumulative_sums_test                        0.9158557883988356 PASS
random_excursion_test                       6.982315183915555e-05 FAIL
random_excursion_variant_test               0.09544277536456673 PASS
```

Gambar 4.3.1 dan **4.3.2** *Seed* dan hasil uji pertama LCG dengan *reseed* menggunakan kamera
Sumber : Dokumen Penulis

```
Seed kamera : 214
Seed kamera : 10
Seed kamera : 23
Seed kamera : 3
Seed kamera : 255
Execution time: 11.713162183761597
```

```
SUMMARY
-------
monobit_test                                0.6478219459340199 PASS
frequency_within_block_test                 2.6741950656167436e-41 FAIL
runs_test                                   0.2626589212692139 PASS
longest_run_ones_in_a_block_test            2.1338534155678214e-203 FAIL
binary_matrix_rank_test                     0.0                FAIL
dft_test                                    0.0                FAIL
non_overlapping_template_matching_test      1.0031591465025078 PASS
overlapping_template_matching_test          0.0                FAIL
maurers_universal_test                      0.0                FAIL
linear_complexity_test                      2.3812900280233977e-28 FAIL
serial_test                                 0.7532461987125562 PASS
approximate_entropy_test                    0.8194043129230172 PASS
cumulative_sums_test                        0.9900682461751418 PASS
random_excursion_test                       0.0                FAIL
random_excursion_variant_test               0.0                FAIL
```

Gambar 4.3.3 dan **4.3.4** *Seed* dan hasil uji kedua LCG dengan *reseed* menggunakan kamera
Sumber : Dokumen Penulis

```
Seed kamera : 149
Seed kamera : 108
Seed kamera : 59
Seed kamera : 83
Seed kamera : 6
Execution time: 11.60648488998413
```

```
SUMMARY
-------
monobit_test                                0.7284495331376171 PASS
frequency_within_block_test                 1.966002281137366e-45 FAIL
runs_test                                   5.78997710444999e-06 FAIL
longest_run_ones_in_a_block_test            3.2559391668230927e-46 FAIL
binary_matrix_rank_test                     0.0                FAIL
dft_test                                    0.0                FAIL
non_overlapping_template_matching_test      1.0499740719967547 PASS
overlapping_template_matching_test          0.0                FAIL
maurers_universal_test                      0.0                FAIL
linear_complexity_test                      0.0                FAIL
serial_test                                 0.0                FAIL
approximate_entropy_test                    0.0                FAIL
cumulative_sums_test                        0.9978143565417876 PASS
random_excursion_test                       0.0                FAIL
random_excursion_variant_test               2.3672220430759696e-199 FAIL
```

Gambar 4.3.5 dan **4.3.6** *Seed* dan hasil uji ketiga LCG dengan *reseed* menggunakan kamera
Sumber : Dokumen Penulis

Hasil dari kamera lebih buruk dibanding LCG memakai sumber waktu maupun tanpa melakukan *reseed*. Dengan sumber ini, LCG rata-rata berhasil melewati 5 jenis uji. Hal ini mungkin disebabkan *seed* yang didapat memberikan hasil yang kurang baik untuk nilai a, b, dan m yang digunakan pada LCG, terutama untuk hasil ketiga yang juga cenderung menurun dibanding hasil uji lain. Kegagalan *serial test* dan *approximate entropy test* pada hasil ketiga menunjukkan bahwa ada pola tertentu yang lebih sering muncul dibanding pola lain. Hal ini juga menjelaskan kegagalan LCG untuk lolos uji non-overlapping template matching test pada hasil pertama. Kemungkinan terdapat dua atau lebih *seed* yang menghasilkan pola barisan yang sama.

4.4. LCG dengan *reseed* dari suara

```
* recording
* done recording
Seed suara : 111
* recording
* done recording
Seed suara : 147
* recording
* done recording
Seed suara : 192
* recording
* done recording
Seed suara : 134
* recording
* done recording
Seed suara : 211
Execution time: 12.35343861579895
```

```
SUMMARY
-------
monobit_test                          0.6468056868299283 PASS
frequency_within_block_test           -4.5422535985082195e-43 FAIL
runs_test                             0.2605556028162373 5 PASS
longest_run_ones_in_a_block_test      2.1338534155678214e-203 FAIL
binary_matrix_rank_test               0.0                FAIL
dft_test                              0.0                FAIL
non_overlapping_template_matching_test 0.9726142771670959 PASS
overlapping_template_matching_test    0.0                FAIL
maurers_universal_test                0.0                FAIL
linear_complexity_test                7.136452965977261e-07 FAIL
serial_test                           7.875203339371796e-13 FAIL
approximate_entropy_test              6.855445908142037e-12 FAIL
cumulative_sums_test                  0.9804795808393527 PASS
random_excursion_test                 0.002532100498915077 FAIL
random_excursion_variant_test         0.06924887875398542 PASS
```

Gambar 4.4.1 dan **4.4.2** *Seed* dan hasil uji pertama LCG dengan *reseed* menggunakan suara
Sumber : Dokumen Penulis

```
* recording
* done recording
Seed suara : 96
* recording
* done recording
Seed suara : 5
* recording
* done recording
Seed suara : 110
* recording
* done recording
Seed suara : 74
* recording
* done recording
Seed suara : 233
Execution time: 12.362784385681152
```

```
SUMMARY
-------
monobit_test                          0.6422406906857108 PASS
frequency_within_block_test           -1.1903054994316785e-43 FAIL
runs_test                             1.6392546460801644e-08 FAIL
longest_run_ones_in_a_block_test      2.1338534155678214e-203 FAIL
binary_matrix_rank_test               0.0                FAIL
dft_test                              0.0                FAIL
non_overlapping_template_matching_test 0.9999999784670915 PASS
overlapping_template_matching_test    0.0                FAIL
maurers_universal_test                0.0                FAIL
linear_complexity_test                0.0                FAIL
serial_test                           0.0                FAIL
approximate_entropy_test              0.0                FAIL
cumulative_sums_test                  0.9910469519872434 PASS
random_excursion_test                 1.0382847291244729e-20 FAIL
random_excursion_variant_test         0.13680821719785347 PASS
```

Gambar 4.4.3 dan **4.4.4** *Seed* dan hasil uji kedua LCG dengan *reseed* menggunakan suara
Sumber : Dokumen Penulis

```
* recording
* done recording
Seed suara : 250
* recording
* done recording
Seed suara : 251
* recording
* done recording
Seed suara : 67
* recording
* done recording
Seed suara : 254
* recording
* done recording
Seed suara : 65
Execution time: 12.460510492324829
```

```
SUMMARY
-------
monobit_test                          0.6483303218539302 PASS
frequency_within_block_test           1.0969850191312589e-39 FAIL
runs_test                             0.2514430449759532 PASS
longest_run_ones_in_a_block_test      2.1338534155678214e-203 FAIL
binary_matrix_rank_test               0.0                FAIL
dft_test                              0.0                FAIL
non_overlapping_template_matching_test 0.7045926711793528 PASS
overlapping_template_matching_test    0.0                FAIL
maurers_universal_test                0.0                FAIL
linear_complexity_test                1.043920706313822e-16 FAIL
serial_test                           0.0008916521179512675 FAIL
approximate_entropy_test              0.003469233107204258 FAIL
cumulative_sums_test                  0.9755710454784623 PASS
random_excursion_test                 5.002842908781697e-30 FAIL
random_excursion_variant_test         0.6901739727756928 PASS
```

Gambar 4.4.5 dan **4.4.6** *Seed* dan hasil uji ketiga LCG dengan *reseed* menggunakan suara
Sumber : Dokumen Penulis

Seed dari suara memiliki hasil yang paling buruk, yaitu hanya melewati 14/3 jenis uji. LCG dengan *seed* suara secara konsisten gagal pada *serial test* dan *approximate entropy test*. Hal ini menunjukkan bahwa LCG cenderung menghasilkan pola-pola tertentu dibanding pola lain. Dari hasil *seed* yang didapat bisa dilihat bahwa *seed* relatif kurang bervariasi. Hal ini mungkin disebabkan suara dari lingkungan memiliki faktor yang tidak acak.

V. Kesimpulan dan Saran

Penggunaan *random seed* dan *reseed* pada LCG bisa memberikan dampak yang berbeda-beda tergantung pada sumber dan perangkat yang digunakan. Salah satu hal yang bisa diperbaiki dengan menggunakan metode tersebut adalah periodisitas (untuk pola yang tidak tumpang tindih). Namun, hasil tetap lebih dipengaruhi oleh *seed* yang didapat dan nilai a dan m dari LCG.

Untuk penelitian berikutnya bisa digunakan uji-uji beserta sumber lain yang memiliki entropi yang lebih baik. Selain itu, untuk sumber kamera dan mikrofon bisa dicoba dengan kondisi lingkungan yang lebih bervariasi.

VI. Ucapan Terima Kasih

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rahmat-Nya dalam penulisan makalah ini sehingga bisa selesai tepat waktu. Penulis juga mengucapkan terima kasih kepada Ibu Fariska Zakhralativa Ruskanda, S.T. M.T. selaku dosen mata kuliah IF2120 Matematika Diskrit kelas K02 karena telah memberikan ilmu untuk menyelesaikan makalah ini dan kepada David Johnston karena telah membuat implementasi uji NIST dalam Python sehingga bisa digunakan untuk analisis. Penulis juga berterima kasih kepada orang tua atas dukungan yang diberikan dalam penulisan makalah.

Referensi

[1] Munir, Rinaldi. 2022. "Teori Bilangan (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian1.pdf (Diakses pada 11 Desember 2022)

[2] Munir, Rinaldi. 2022. "Teori Bilangan (Bagian 2)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian2.pdf (Diakses pada 11 Desember 2022)

[3] Munir, Rinaldi. 2022. "Teori Bilangan (Bagian 3)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Teori-Bilangan-2020-Bagian3.pdf (Diakses pada 11 Desember 2022)

[4] Munir, Rinaldi. 2022. "Rekursi dan Relasi Rekurens (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Rekursi-dan-relasi-rekurens-(Bagian-1).pdf (Diakses pada 11 Desember 2022)

[5] https://github.com/dj-on-github/sp800_22_tests. Diakses pada 11 Desember 2022.

[6] National Institute of Standards and Technology. 2010. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications". https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf (Diakses pada 11 Desember 2022)

[7] http://people.csail.mit.edu/hubert/pyaudio/. Diakses pada 11 Desember 2022.

[8] https://1library.net/article/entropy-sources-recommendations-random-numbers-generation-randomness-require.qol1r2jq. Diakses pada 11 Desember 2022.

[9] https://www.oreilly.com/library/view/secure-programming-cookbook/0596003943/ch11s06.html. Diakses pada 11 Desember 2022.

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, 11 Desember 2022

Nathan Tenka
13521172