Perancangan Pseudo Random Number Generator Berbasis Julia Set

Léa Angelina (13506117)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
xcoldblizzard@itb.ac.id

ABSTRAKSI

Pseudorandom Number Generator (PRNG) atau Pembangkit Bilangan Acak Semu adalah algoritma untuk membangkitkan sebuah seri bilangan yang propertinya mendekati bilangan acak, akan tetapi tidak benar-benar acak karena pada titik tertentu dapat berulang secara periodik. Namun demikian, Pembangkit Bilangan Acak Semu pada umumnya memiliki periode yang cukup panjang untuk membuat seri tersebut cukup acak untuk aplikasi praktis. PRNG memiliki banyak sekali aplikasi, yaitu dalam kriptografi, pembuatan game, pembangkitan sample data, dan lain-lain. Dalam makalah ini akan dirancang sebuah skema Pembangkit Bilangan Acak Semu yang berbasis Julia Set. Julia Set adalah sebuah set yang merupakan subset dari Mandelbrot Set. Mandelbrot set adalah sebuah kumpulan titik tertentu yang sisi-sisinya membangkitkan bentuk-bentuk pola fraktal. Julia set dipilih karena memiliki banyak properti *chaos*, yang baik untuk membangkitkan bilangan acak semu.

Kata Kunci: Chaos, Fractal, Random, PRNG, Julia Set, Mandelbrot Set, Fatou Dust

I. PENDAHULUAN

Sulit membayangkan aplikasi kriptografi yang tidak memanfaatkan bilangan acak. Sebut saja kunci *session*, initialization vectors, *salt* untuk di*hash* dengan password, parameter unik pada operasi tanda tangan digital, dan *nonces* (bilangan acak yang hanya dipakai sekali) pada protokol, semuanya diasumsikan sebagai bilangan acak oleh perancang sistem. Sayangnya, banyak aplikasi kriptografi tidak memiliki sumber bilangan acak yang bisa dipercaya performansinya. Akan tetapi, mereka memakai sebuah mekanisme kriptografi yang bernama *Pseudo-Random Number Generator* (PRNG) atau pembangkit bilangan acak semu untuk membangkitkan nilai-nilai acak yang dibutuhkan tersebut.

Akan tetapi, *Pseudo-Random Number Generator* (PRNG) atau pembangkit bilangan acak semu bukannya tidak memiliki masalah. Masalah utama yang dihadapi adalah bilangan-bilangan yang dibangkitkan tidak cukup acak, karena itu pembangkit bilangan acak semu harus memiliki algoritma yang baik yang memiliki basis matematika yang kuat yang menjamin bahwa sekuens bilangan yang dibangkitkannya cukup acak dan tidak bisa diprediksi. Masalah lain yang dihadapi adalah berulangnya seri bilangan secara periodik pada titik tertentu yang memang merupakan sifat dari pembangkit bilangan acak semu. Untuk menanggulanginya, pembangkit bilangan acak semu harus memiliki algoritma yang menjamin periode yang dimilikinya cukup panjang, dan pengguna dapat membangkitkan bilangan acak sebanyak penggunaan yang mungkin untuk semua keperluan praktis sebelum periodenya terulang.

Mempertimbangkan faktor-faktor di atas, dipilihlah sebuah skema untuk membangkitkan bilangan acak semu yang berbasiskan fraktal, tepatnya Julia set, karena fraktal adalah pola yang lazim ditemui di alam, di mana fenomena yang ditemui di alam bersifat *chaotic* dan terlihat acak, tidak terpola, dan sangat kompleks.

II. PEMBANGKIT BILANGAN ACAK SEMU

Pseudorandom Number Generator (PRNG) atau Pembangkit Bilangan Acak Semu, atau disebut juga sebagai *Deterministic Random Bit Generator* (DRBG) atau Pembangkit Bit Acak Deterministik, adalah sebuah algoritma untuk membangkitkan sebuah seri bilangan yang memperkirakan properti dari bilangan acak. Sekuensnya tidak benar-benar acak karena seluruhnya ditentukan oleh suatu set yang relatif kecil yang berisi nilai-nilai awal, yang disebut state PRNG. Walaupun sekuens yang benar-benar mendekati acak dapat dibangkitkan dengan pembangkit bilangan acak berdasarkan hardware, bilangan acak semu memegang peranan penting dalam simulasi (contoh: sistem fisik dengan metode Monte Carlo), dan memegang peranan utama dalam aplikasi kriptografi dan generasi prosedural.

Kelas-kelas umum dari algoritma-algoritma tersebut adalah generator linear kongruen, generator fibonacci lagged, register shift umpan-balik linear, umpan-balik dengan register carry shift, dan register shift umpan-balik yang digeneralisasi. Contoh baru dari algoritma bilangan acak semu yaitu Blum Blum Shub, Fortuna, dan Twister Mersenne. Analisa matematika yang mendalam

dibutuhkan untuk memberi rasa yakin bahwa sebuah pembangkit bilangan acak semu membangkitkan bilangan-bilangan yang cukup acak untuk memenuhi kebutuhan pengguna.

III. JULIA SET

Julia set ditemukan oleh seorang matematikawan asal Perancis bernama Gaston Julia yang meneliti tentang fraktal pada tahun 1915. Julia mempelajari ekspresi polinomial rasional pada derajat yang bervariasi (contoh: $z^4 + z^3/(z + 1) + z^2/(z^3 + 4z^2 + 5) + c$), tetapi bentuk paling umumnya adalah $f(z) = z^2 + c$. Di sini z merepresentasikan sebuah variabel yang berbentuk a+ib (a dan b adalah bilangan real) yang bisa juga termasuk semua nilai pada bidang kompleks. Kuantitas c juga didefinisikan sebagai bilangan kompleks, tetapi untuk setiap Julia set yang diberikan, bilangan c adalah konstan (maka itu c disebut juga sebagai parameter). Dengan kata lain, terdapat Julia set sejumlah tak-berhingga, yang masing-masing memiliki nilai c sendiri. Julia set dengan nilai c yang kecil (misal: $|c| < \sim 2$) memiliki grafik yang menarik.

Jika digunakan sekali, ekspresi sederhana $f(z)= z^2 + c$ hanya memiliki kemungkinan kecil untuk menciptakan sesuatu yang menarik—hanya dengan mengiterasinya secara berulanglah persamaan tersebut dapat menghasilkan sebuah Julia set. Jika hasil keluaran dari ekspresi f(z) digunakan sebagai feedback ke dalam persamaan tersebut sebagai nilai baru dari z, barulah hal tersebut disebut sebagai iterasi, semacam proses umpan-balik. Maka, untuk nilai n sembarang:

$$z_{n+1} = f(z) = z_n^2 + c$$

dan setiap nilai baru f(z) dari hasil perhitungan menjadi nilai masukan selanjutnya untuk z melalui perulangan umpan-balik.

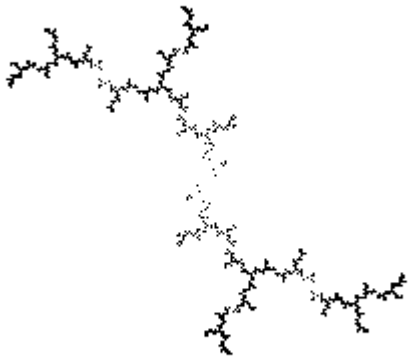
Untuk kasus khusus di mana c=0+0i, Julia set hanya berbentuk lingkaran (bukan fraktal) dengan jari-jari 1. Untuk setiap z, transformasi ini terdiri dari sebuah kontraksi (untuk $|z|<1$) atau dilatasi (untuk $|z|>1$) yang berasal dari perkalian dengan $|z|$, dan juga perkalian sudut polar z dengan dua, lalu ditranslasi oleh c.

Untuk setiap nilai awal z yang diberikan, misalnya z_0, terdapat dua kemungkinan apa yang akan terjadi terhadap nilai f(z) yang diiterasi pada saat nilai n bertambah mendekati tak terhingga:antara f(z) akan terus bertambah tanpa batas, atau akan tetap berada dalam batas. Titik z_0 dalam bidang kompleks yang terus bertambah (tidak berada dalam batas-batas) dengan iterasi f(z) yang terus menerus disebut berada dalam *escape set* E_c. Seluruh titik lainnya yang berada dalam bidang kompleks tetap berada di dalam batas walaupun nilai n terus bertambah mendekati nilai tak hingga memiliki istilah "tawanan" (prisoner) dan disebut berada dalam set tawanan P_c yang didefinisikan untuk sebuah nilai c yang diberikan.

Semua titik harus berada di satu set atau di set lainnya. Batasan yang umum antara set escape dan set tawanan disebut set Julia J_c, didefinisikan untuk nilai tertentu dari c. Jari-jari ambang batas $r(c) = \max(|c|, 2)$ menyediakan kriteria tes yang berguna untuk implementasi melalui komputer. Jika sebuah orbit z_k melebihi jari-jari ambang batas r(c), maka sudah pasti bahwa orbitnya akan keluar dari batas dan mendekati nilai tak hingga dan maka itu dapat disimpulkan bahwa titik awal adalah set escape.

Terdapat sedikit ambiguitas pada masalah definisi dalam literatur, apakah titik batas (Contoh: set Julia) adalah termasuk dalam set tawanan, dan ini harus diperhatikan. Bidang kompleks hanya dibagi menjadi dua, yaitu bagian set tawanan dan bagian set escaping, jadi titik batasnya harus merupakan bagian dari set tawanan karena titik batas tersebut tidak bisa merupakan bagian dari set escaping (atau jika tidak, titik-titik tersebut akan terlepas dari batas jika diiterasi secara berulang, dan itu bukan merupakan kasus yang diamati).

Selain itu, terdapat juga set Julia (contoh: yang terdiri dari titik-titik batas) yang tidak berada di sekitar titik-titik tawanan dalam. Karena definisi set tawanan adalah anggota dari bidang kompleks setelah set escape dikeluarkan, batasan dan titik-titik tawanan harus sama. Contoh dari set tersebut adalah J_c untuk c=0+i



Gambar 1. Set Julia untuk nilai c=0+1i

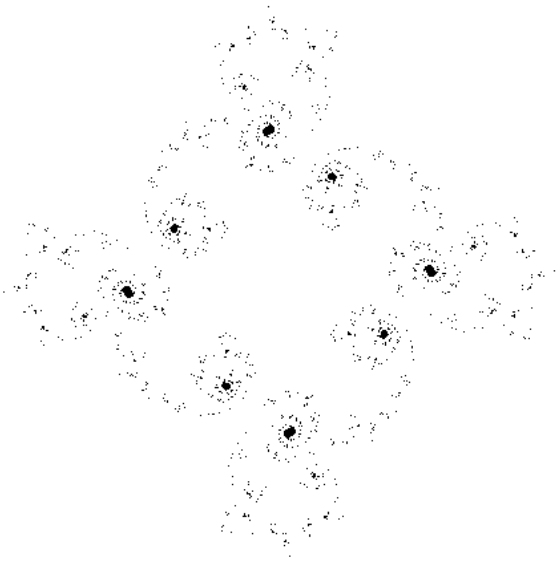
Tergantung dari nilai c yang dipilih, set Julia yang dihasilkan bisa saling terhubung maupun tidak terhubung—faktanya, antara benar-benar terhubung ataupun sama sekali tidak terhubung. Terdapat beberapa tipe keterhubungan secara matematis. Pada kasus set Julia, tipe keterhubungan yang dimaksud adalah keterhubungan jalur, yang berarti jalur dapat ditelusuri dari satu titik pada set ke titik-titik lain dalam set tanpa meninggalkan set. Set

julia yang saling terhubung adalah "terhubung secara keseluruhan", tidak hanya terhubung secara lokal, dari sebuah hasil yang ditunjukkan secara independen oleh Julia dan fatou. Secara topologis, set Julia yang saling terhubung adalah antara ekuivalen dengan lingkaran yang sangat terdeformasi atau dengan sebuah kurva dengan jumlah cabang dan sub-cabang yang tak berhingga yang disebut dengan dendrite (contoh: Julia set untuk c=0+i, seperti pada gambar di halaman sebelumnya).

Mandelbrot menyebut set Julia yang tidak terhubung sebagai "debu" dari kumpulan titik, atau "debu Fatou" (berasal dari Pierre Fatou 1878-1929). Istilah ini logis, karena set Julia yang saling tidak terhubung terdiri dari titik-titik individual pada bidang kompleks, yang terlihat seperti debu yang disebarkan pada suatu bidang, tidak terhubung satu sama lain. Istilah lain yang digunakan untuk mendeskripsikannya adalah debu Cantor.

Set Cantor merupakan sebuah set yang sama sekali tidak terhubung yang dihasilkan oleh proses membagi segmen garis [0,1] menjadi tiga bagian dan membuang segmen tengan, menghasilkan [0, 1/3] dan [2/3, 1], lalu mengulang proses itu kembali untuk tiap segmen garis yang tersisa mendekati tak hingga.

Distribusi dari titik-titik pada set Julia yang saling tidak terhubung secara kualitatif menyerupai tampilan dari debu Cantor karena mereka sama-sama benar-benar tidak terhubung, setiap titik terlepas satu sama lain. Set yang saling tidak terhubung adalah benar-benar tidak terhubung sama sekali, yang berbentuk kumpulan tak terhingga dari titik-titik yang saling terisolasi satu sama lain.

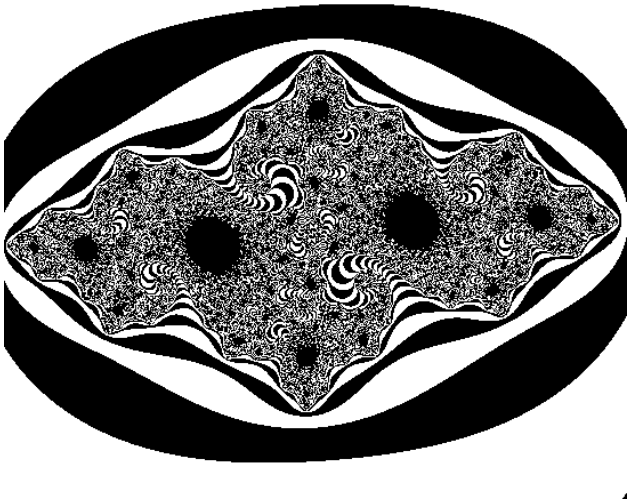


Gambar 2. Disconnected Julia Set

Gambar di atas adalah contoh yang dihasilkan pada persamaan $f(z)=z^4-c$, c=0.8+0.09i.

IV. DESKRIPSI ALGORITMA BARU

Pada dasarnya, algoritma pembangkit bilangan acak semu ini memanfaatkan set Julia dalam membangkitkan bilangan-bilangan acak, karena set Julia memiliki tingkat keacakan yang tinggi, tetapi bisa juga tidak. Hal itu sangat ditentukan oleh persamaan yang dipilih dan nilai c yang dimasukkan. Tingkat keacakan bisa jadi sangat berbeda walaupun perbedaan nilai c yang dipilih kecil. Seperti itu pula sifat grafik pada set Julia. Perbedaan kecil dapat menghasilkan grafik dengan kompleksitas yang berbeda, seperti pada prinsip dasar teori chaos, yang dapat menghasilkan perbedaan hasil yang sangat besar walaupun perubahan keadaan awal yang dilakukan pada sebuah sistem non-linear sangat kecil, karena itulah chaos disebut sebagai sensitif terhadap keadaan awal, atau sering digambarkan melalui ekspresi *"the presence or absence of a butterfly flapping its wings could lead to creation or absence of a hurricane*" karena properti kesensitifan itulah teori chaos menjadi kandidat yang baik untuk membuat sebuah Pembangkit Bilangan Acak Semu. Penerapannya misalnya pada grafik di bawah ini:

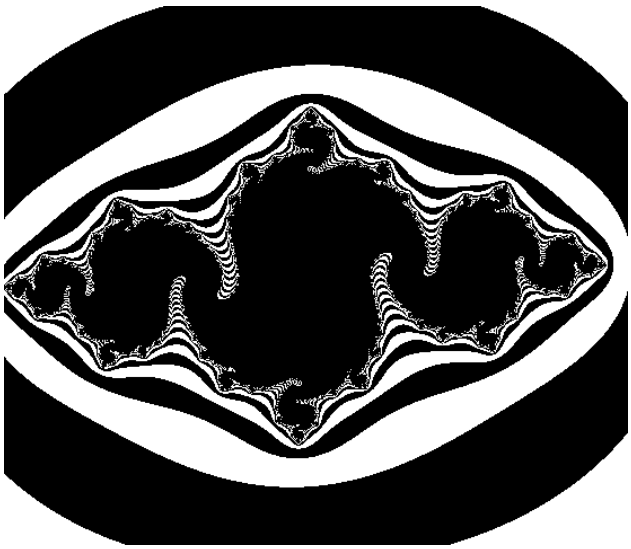


Gambar 3- Set Julia untuk c=0.745+0.113i

Gambar di atas memiliki persamaan yang sangat sederhana, yaitu

$$z_{n+1} = f(z) = z_n^2 + c$$

namun menghasilkan bentuk grafik dengan kompleksitas tinggi seperti itu. Hal itulah yang menjadi keunikan set Julia terutama dengan bilangan kompleks, bandingkan gambar di atas dengan gambar di bawah ini:



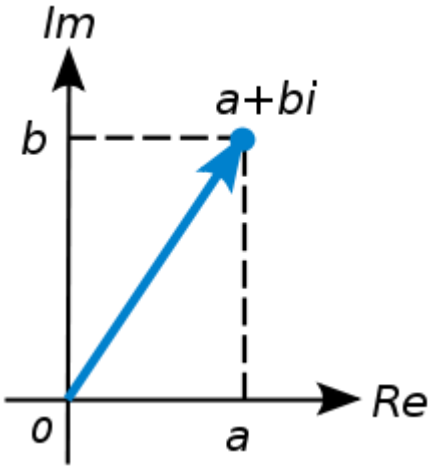
Gambar 4- Set Julia untuk c=0.79+0.110i

Pada pilihan nilai c yang kedua, grafik yang dihasilkan jauh tidak sekompleks grafik yang dihasilkan pilihan nilai c pertama, walaupun perbedaan nilai c antara kedua grafik itu kecil sekali.

Karena hal di atas, maka perlu dipilih persamaan tertentu dan nilai c tertentu yang menghasilkan pola sekompleks mungkin.

Operasi akan dilakukan pada bilangan kompleks, karena set Julia dengan bilangan kompleks menghasilkan pola yang lebih kompleks dan tidak terprediksi dibandingkan set Julia dengan bilangan bulat.

Bilangan kompleks adalah sebuah bilangan yang terdiri dari dua bagian, yaitu bagian real dan bagian imajiner. Bilangan kompleks memperluas konsep dari garis bilangan satu dimensi ke bidang bilangan dua dimensi dengan menggunakan garis bilangan untuk bagian real dan menambahkan sumbu vertikal untuk menggambarkan bagian imajiner. Dengan cara ini, bilangan kompleks memperluas masalah yang dapat diselesaikan karena bilangan kompleks mengandung bilangan real sehingga dapat menyelesaikan masalah bilangan real, tetapi juga memiliki perluasan berupa imajiner sehingga dapat menyelesaikan masalah-masalah yang tidak dapat diselesaikan hanya dengan menggunakan bilangan real.



Gambar 5. Grafik dari bilangan kompleks

Bilangan kompleks dapat direpresentasi secara visual sebagai pasangan dari angka-angka yang membentuk vektor dari sebuah diagram yang disebut diagram Argand, yang merepresentasikan bidang kompleks. Re adalah sumbu real, Im adalah sumbu imajiner, dan i adalah lambang imajiner, yaitu akar dari -1.

Bilangan kompleks memiliki operasi-operasi aritmatika yang berbeda dengan bilangan biasa. Pada penjumlahan dan pengurangan, operasinya cukup intuitif, yaitu:

Penjumlahan: z=x+yi, ditambah dengan c=a+bi, maka
$z + c = (x + a)+(y + b)i$.

Pada pengurangan, serupa dengan yang di atas.

$z – c = (x - a) + (y - b)i$.

Tetapi pada perkalian dan pembagian, operasinya lebih kompleks dan memungkinkan perolehan nilai-nilai yang bervariasi, karena operasi bilangan kompleks terutama pada perkalian, pembagian, perpangkatan, trigonometri, dan lain sebagainya dilakukan pada bidang kompleks, bukan pada garis bilangan seperti bilangan biasa.

Pada perkalian, aturannya secara umum adalah sebagai berikut:

$(x + yi)(u + vi) = (xu – yv) + (xv + yu)i$.

Sedangkan pada pembagian, operasi yang dilakukan adalah sebagai berikut:

$$\frac{x + yi}{u + vi} = \frac{(xu + yv) + (–xv + yu)i}{u^2 + v^2}$$

Pada operasi di atas, dapat dilihat bahwa operasinya memungkinkan variasi yang cukup luas, ditambah dengan fakta bahwa nilai z_i dimasukkan sebagai input untuk nilai

z_{i+1}, nilai z_{i+1} dimasukkan sebagai input untuk nilai z_{i+2} dan seterusnya.

Karena pada beberapa persamaan nilai z yang dihasilkan melepaskan diri dari titik batas dan mendekati tak hingga (set escape), maka harus dipikirkan bagaimana mentransformasikan nilai z itu menjadi berada di dalam batas masukan yang didefinisikan user, pada hal ini, dipilih penggunaan modulus nilai maksimum, karena metode itu adalah metode yang paling sederhana namun efektif dalam membatasi nilai yang dihasilkan.

V. IMPLEMENTASI DAN ANALISIS

Sebagai fungsi antara, dibuat suatu fungsi modulus yang menangani masukan berupa float dan menghasilkan keluaran berupa float juga, karena fungsi modulus yang biasa hanya menghasilkan bilangan bulat. Tetapi jika terlalu sering dilakukan pembulatan, nilai yang dihasilkan akan menjadi sederhana karena cakupannya lebih terbatas. Fungsi modulus digunakan dalam frekuensi yang cukup besar dalam algoritma ini, karena setiap saat harus dijamin bahwa nilai-nilai yang dihasilkan oleh setiap operasi aritmatika tidak melebihi batas yang dimiliki oleh tipe data float, atau jika tidak akan dihasilkan nilai tak hingga (infinity), atau tak terdefinisi sehingga operasi tidak dapat dilanjutkan dan berhenti di tengah (jika z_i adalah tak hingga, maka nilai z_{i+1} tidak dapat dihasilkan karena masukan pada z_{i+1} adalah z_i yang merupakan nilai tak hingga sehingga tidak dapat digunakan pada operasi. Dan jika z_{i+1} tidak memiliki nilai, maka nilai-nilai yang seterusnya juga tidak dapat dihasilkan karena tidak adanya nilai input yang bisa digunakan).

```
float mf(float f1, float f2)
        {
            int result = (int)(f1 / f2);
            float mod = f1 - (int)(result) *
f2;
            return mod;
        }
```

Snippet 1. Fungsi modulus untuk masukan float yang menghasilkan nilai float

Pada percobaan kali ini, persamaan yang digunakan adalah

$F(z)=z^2-z+c$

Karena itu dibutuhkan juga suatu fungsi antara yang berupa perpangkatan dua dari sebuah bilangan kompleks yang diimplementasikan sebagai berikut:

```
float[] powtwo(float x1, float x2)
        {
            float[] r=new float[2];
            r[0] = ((x1 * x1)%Int32.MaxValue)
- ((x2 * x2)%Int32.MaxValue);
            r[1] = (2 * (x1 * x2)) %
Int32.MaxValue;
            return r;
        }
```

Snippet 2. Fungsi perpangkatan dua bilangan kompleks

Sedangkan implementasi rumusnya,

```
void PRNG(int numgen, int min, int max, int
seed, float l1, float l2)
        {
            float z1, z2, m;
            float[] fz=new float[2];
            Random r=new Random(seed);

z1=mf((r.Next()*(float)r.NextDouble()),
Int32.MaxValue);
            z2 =mf((r.Next() *
(float)r.NextDouble()), Int32.MaxValue);

            for (int i = 0; i < numgen; i++)
            {
                fz = powtwo(z1, z2);
                fz[0] =mf(((fz[0] + l1)-z1),
Int32.MaxValue);
                fz[1] =mf(((fz[1] + l2)-z2),
Int32.MaxValue);
                m = ((fz[0] + fz[1]) * (fz[0]
- fz[1])) % max;
                if (m < 0)
                {
                    m = m * -1;
                }
                Console.WriteLine(m);
                z1 =mf(fz[0],
Int32.MaxValue);
                z2 =mf(fz[1],
Int32.MaxValue);
            }
            Console.ReadKey();
        }
```

Snippet 3. Fungsi Pembangkit Bilangan Acak Semu dengan persamaan $F(z)=z^2-z+c$

Source code di atas menggunakan fungsi random default compiler untuk membangkitkan nilai acak sebagai koordinat awal z. Pada umumnya tiap PRNG membutuhkan seed yang baik untuk menghasilkan deretan angka yang cukup acak, untuk mencegah nilai awal z yang kurang baik atau terlalu kecil (misal, jika c mendekati nol), maka dilakukan pembangkitan melalui fungsi random yang dimiliki compiler, akan tetapi hanya untuk pembangkitan nilai awal untuk mencegah mudah diprediksinya bilangan pertama.

Hasilnya adalah,

```
How many numbers you'd like to generate?
25
Enter the minimum and maximum boundary:
-1000
1000
Enter seed:
97432
Enter real and imaginary part:
0.934
0.6
192
-720
-968
120
920
0
120
-352
272
480
256
328
872
-584
600
-256
272
24
648
568
-96
936
24
-320
-872
Time Elapsed= 0,02734375
```

Nilai c=0.934+0.6i dipilih di sini karena grafiknya membentuk set yang disconnected (debu fatou). Seperti yang ditunjukkan di halaman berikut ini:

Gambar 6. Debu Fatou yang dihasilkan oleh F(z)=z2-z+c, di mana c=0.934+0.6i

Perhatikan ketidakterhubungan antara elemen-elemen pada gambar di atas.

Pada tes kecepatan, untuk membangkitkan 500,000 bilangan 16-bit dibutuhkan waktu 169.3262 detik.

Ketika algoritma ini diujicoba menggunakan Entropy test, hasilnya:

Entropy = 7.999183 bits per byte.

Optimum compression would reduce the size of this 62237 character file by 0 percent

Chi square distribution for 62237 samples is 7512.49, and randomly would exceed this value less than 0.01 percent of the times.

Arithmetic mean value of data bytes is 127.1362 (127.5 = random). Monte Carlo value for Pi is 3.141791622 (error 0.10 percent). Serial correlation coefficient is 0.000156 (totally uncorrelated = 0.0)

Dengan kata lain, performa PRNG ini dapat dibilang baik dengan tingkat keacakan tinggi.

Bandingkan nilai chi-squarenya dengan fungsi `rand()` standar UNIX 8 bit:

Chi square distribution for 500000 samples is 0.01, and randomly would exceed this value more than 99.99 percent of the times.

Maka performa algoritma ini jauh melampaui fungsi sederhana `rand()`.

V. KESIMPULAN

Pembangkit Bilangan Acak Semu berbasis Chaos (set Mandelbrot, set Julia) secara umum memiliki performa yang tinggi namun sangat tergantung pada algoritma dan nilai c yang dipilih. Performa keacakannya baik karena fraktal menggambarkan pola-pola yang ada di alam yang bersifat acak dan tidak terprediksi. Akan tetapi baru sempat dilakukan tes yang menguji apakah algoritma ini aman secara kriptografis atau tidak, yaitu tes entrophi (oleh John Walker) jadi sementara algoritma ini tidak akan diaplikasikan terlebih dahulu pada aplikasi-aplikasi yang menuntut keamanan yang tinggi pada pembangkit bilangan acak semu melainkan diaplikasikan untuk *general purpose* terlebih dahulu sampai dilakukan tes yang membuktikan keamanan algoritma ini, walaupun hasil tes ENTnya cukup baik, karena penulis belum merasa dapat menjamin keamanan algoritma ini sebelum tes-tes yang lain juga dilakukan.

DAFTAR PUSTAKA

[1] http://en.wikipedia.org/wiki/Pseudorandom_number_generator/

[2] http://en.wikipedia.org/wiki/Julia_set
[3] http://www.fourmilab.ch/random/
[4] http://www.clarku.edu/~djoyce/complex/mult.html
[5] http://en.wikibooks.org/wiki/Fractals/Iterations_in_the_complex_plane/Julia_set
[6] http://www.chaospro.de/julmand.php
[7] http://dotnet.jku.at/applications/course03/Braune/
http://www.mcgoodwin.net/julia/juliajewels.html

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, 9 Mei 2010

ttd

Léa Angelina
13506117