Implementasi Fungsi Hash untuk Pembangkitan Bilangan Acak Semu

Hani'ah Wafa - 13516053
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
haniahwafa@gmail.com

***Abstract*—Bilangan acak sangat berguna terutama dalam sistem keamanan komputer. Fungsi hash yang dapat memberikan nilai yang acak dan nilai yang berbeda meskipun untuk seed yang relatif sama dapat memberikan kontribusi dalam pembuatan algoritma pembangkit bilangan acak semu.**

***Keywords*—Bilangan acak semu, pembangkit bilangan acak, fungsi hash**

I. Pendahuluan

Bilangan acak sangat penting dalam melakukan analisa statistik dan dalam teori probabilitas. Bilangan acak juga banyak sekali digunakan untuk *gambling*, memberikan hasil yang tidak dapat diprediksi pada permainan di komputer dan sangat penting dalam bidang kriptografi.

Dalam kriptografi atau keamanan sistem dibutuhkan bilangan yang tidak dapat ditebak oleh penyerang. Dapat dikatakan bahwa bilangan acak yang baik adalah hal mendasar bagi hampir semua sistem keamanan komputer. Tentu saja kita tidak bisa menggunakan bilangan yang sama secara terus menerus. Sehingga dalam hal ini kita ingin menghasilkan suatu bilangan dengan cara yang tidak dapat diprediksi sehingga penyerang tidak dapat menebaknya.

Seiring dengan perkembangan teknologi dan meningkatnya kesadaran akan pentingnya bilangan acak bagi sistem keamanan komputer, terdapat banyak sekali algoritma atau *library* yang menyediakan layanan untuk menghasilkan bilangan acak seperti *library* yang disediakan oleh bahasa pemrograman seperti python, cpp dan lain-lain.

Sebagai fungsi yang diharapkan memiliki *minimal collision*, fungsi hash mungkin akan memberi kontribusi positif dalam pembangkitan bilangan acak. Dengan properti fungsi hash yang menghasilkan nilai yang acak dan berbeda secara signifikan meskipun untuk *seed* yang hanya berbeda satu karakter, kita juga dapat berharap dapat membangkitkan bilangan yang acak dari modifikasi nilai fungsi hash yang kita peroleh dari suatu *seed*.

II. Dasar teori

A. Bilangan Acak

Bilangan acak adalah urutan atau rangkaian bilangan yang memenuhi dua kondisi sebagai berikut.

1. Memiliki nilai yang secara uniform terdistribusi pada suatu interval
2. Tidak mungkin untuk memprediksi nilai bilangan acak selanjutnya berdasarkan nilai saat ini atau nilai yang terdahulu.

B. Pembangkit Bilangan Acak

Pembangkit bilangan acak atau *random number generator* adalah teknologi yang didesain untuk membangkitkan himpunan bilangan acak yang tidak boleh memperlihatkan pola yang dapat dibedakan. Pembangkit bilangan acak dibedakan menjadi dua jenis sebagai berikut.

1. *Pseudo-Random Number Generator (PRNGs)*

   Pseudo-random number generator atau pembangkit bilangan acak semu adalah algoritma yang menggunakan formula matematika dalam melakukan perhitungan untuk menghasilkan rangkaian bilangan yang tampak acak.

   PRNGs bersifat efisien atau dengan kata lain dapat menghasilkan banyak bilangan dengan waktu yang singkat. Selain efisien *PRNGs* juga bersifat *deterministic* yang artinya rangkaian bilangan yang diberikan atau dihasilkan dapat diproduksi ulang di waktu lain jika nilai awal dari rangkaian bilangan tersebut diketahui. *PRNGs* juga bersifat periodik yang artinya pada suatu waktu tertentu nilai yang dihasilkan oleh algoritma ini akan berulang.

   PRNGs berguna untuk aplikasi yang membutuhkan banyak bilangan acak dan membutuhkan agar rangkaian bilangan yang sama dapat diproduksi ulang di lain waktu.
2. *True Random Number Generator (TRNGs)*

   Dibandingkan dengan *PRNGs* yang menggunakan fungsi atau kalkulasi matematis, *TRNGs* akan mengekstrak nilai ketidakaturan dari fenomena fisik

dan memperkenalkannya ke komputer. Fenomena fisik yang dimaksud disini salah satu contohnya adalah pergerakan *mouse* atau rentang waktu antar tombol pada *keyboard* ditekan.

Contoh lain yang akan bagus digunakan untuk *TRNGs* adalah dengan menggunakan peluruhan radioaktif. Peluruhan radioaktif ini tidak dapat diprediksi namun dapat dengan mudah dideteksi oleh komputer.

Karakteristik *TRNGs* sangat berbeda dengan karakteristik *PRNGs*. *TRNGs* bersifat lebih tidak efisien jika dibanding dengan *PRNGs*. *TRNGs* akan membutuhkan waktu yang lebih lama untuk dapat menghasilkan bilangan. *TRNGs* juga bersifat *non-deterministic* yang berarti rangkaian bilangan yang pernah dihasilkan tidak dapat diproduksi lagi meskipun beberapa rangkaian bilangan tentu saja mungkin untuk muncul beberapa kali. Karakteristik lainnya dari *TRNGs* adalah aperiodik.

Karakteristik yang dimiliki *TRNGs* ini secara kasar membuat *TRNGs* cocok digunakan untuk aplikasi yang tidak cocok dengan penggunaan *PRNGs*. Contoh aplikasi yang akan baik jika menggunakan TRNGs adalah aplikasi yang digunakan untuk melakukan enkripsi data, *gambling* dan juga dalam permainan.

C. Hash Function

Fungsi hash adalah metode komputasi yang dapat memetakan data dengan ukuran yang tidak ditentukan ke data dengan ukuran yang sudah ditetapkan dan biasanya lebih pendek dari ukuran aslinya.

Contoh penggunaan *hashing* adalah untuk pemberian index dan peletakan suatu *item* di *database*. Hal tersebut dikarenakan akan lebih mudah untuk melakukan pencarian terhadap nilai hash yang lebih pendek dibandingkan dengan melakukan pencarian terhadap nilai string asli yang panjang.

Selain dalam *database*, *hashing* juga digunakan dalam bidang kriptografi untuk menghasilkan *message digest*. Fungsi tersebut menggunakan fungsi matematika yang bersifat *one-way*. Pada fungsi *one-way* artinya kita akan sangat sulit atau bahkan tidak mungkin untuk mengkalkulasi nilai input dari nilai hash yang dihasilkan kecuali jika algoritma fungsi hash-nya diketahui.

D. Randomness Test

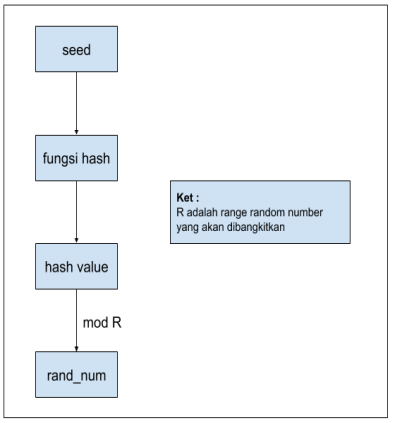
Randomness test atau tes ketidakteraturan adalah tes yang digunakan untuk menganalisis distribusi dari suatu himpunan data dan menentukan apakah himpunan data tersebut adalah acak atau tidak memiliki pola.

Beberapa cara sederhana untuk melakukan tes terhadap ketidakteraturan adalah dengan melihat frekuensi kemunculan suatu nilai dan melihat urutan kemunculan nilai.

IV. Implementasi algoritma

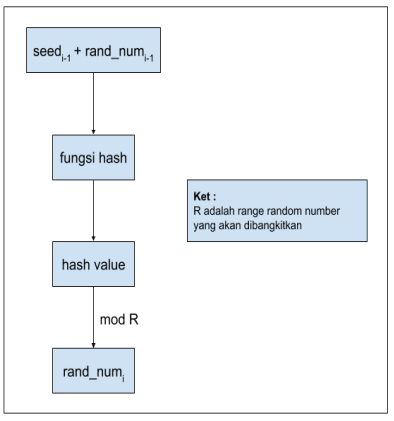
Untuk menghasilkan bilangan acak dengan menggunakan algoritma dalam makalah ini, pengguna akan diminta untuk memasukkan tiga nilai sebagai berikut.

- N : merupakan banyaknya bilangan acak semu yang ingin dibangkitkan
- S : merupakan *seed* yang akan digunakan untuk membangkitkan bilangan acak semu. Seed yang digunakan berupa string.
- R : merupakan range bilangan acak semu yang akan dibangkitkan. Dimana nilai bilangan acak yang akan dihasilkan memenuhi -1 < bilangan acak < R



Gambar 2.1. Bilangan acak pertama

Untuk menghasilkan bilangan acak yang pertama, nilai seed dari masukan pengguna akan dijadikan parameter dalam pemanggilan fungsi hash. Pada algoritma ini fungsi hash yang digunakan adalah fungsi hash md5. Setelah dilakukan pemanggilan terhadap fungsi hash, maka kita akan memiliki hash value yang direpresentasikan dalam bentuk hexadecimal. Langkah terakhir untuk menghasilkan bilangan acak semu dalam algoritma ini adalah dengan mencari nilai sisa pembagian dari hash value dengan nilai R yang telah dimasukkan oleh pengguna.



Gambar 2.2. Bilangan acak kedua dan seterusnya

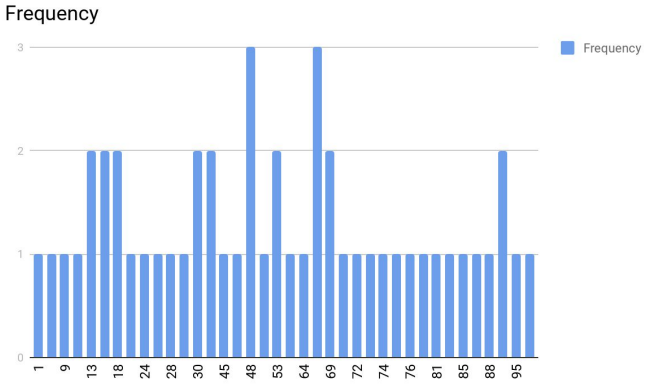
Jika dibandingkan dengan pembangkitan bilangan acak yang pertama dalam pembangkitan bilangan acak yang kedua dan seterusnya hanya ada perbedaan pada *seed* yang digunakan saja. *Seed* yang akan digunakan merupakan hasil *concatination* dari *seed* yang digunakan dalam pembangkitan bilangan acak sebelumya dengan hasil bilangan acak yang terakhir dibangkitkan. Setelah itu proses selanjutnya tidak berbeda dengan pembangkitan bilangan acak yang pertama, yaitu melakukan pemanggilan terhadap fungsi hash dengan parameter masukkan adalah *seed*. Kemudian mencari nilai sisa pembagian dari hash value yang dihasilkan dengan nilai R yang dimasukkan oleh pengguna.

V. Simulasi dan Pembahasan Hasil

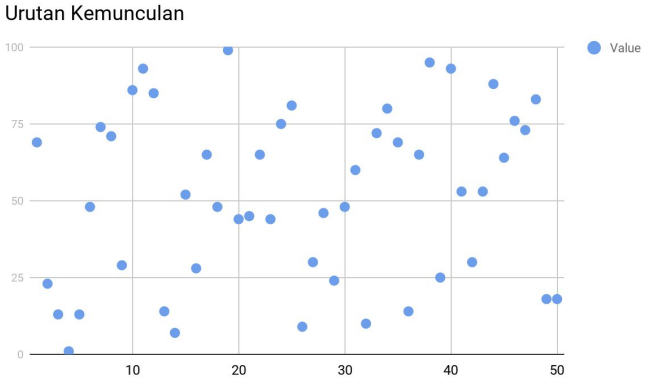
Pada bagian ini akan dipaparkan hasil implementasi algoritma yang telah dijelaskan di bagian sebelumnya. Agar nilai bilangan acak yang dihasilkan dapat terlihat dengan jelas, maka dalam eksperimen ini akan digunakan nilai N dan R yang tidak terlalu besar.

1. **Eksperimen 1**
   N : 50
   S : sebentar lagi liburan
   R : 101

   Hasil eksperimen



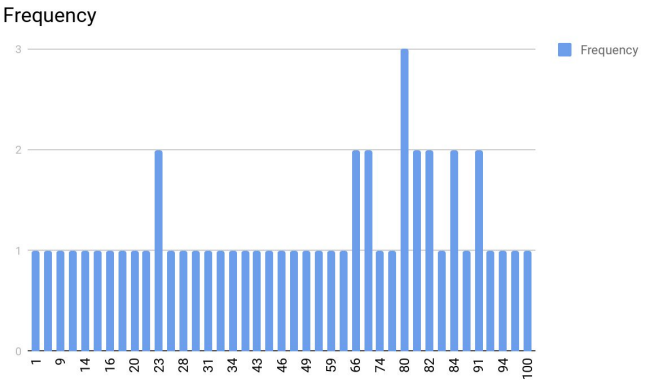
Gambar 5.1.1 Frekuensi kemunculan



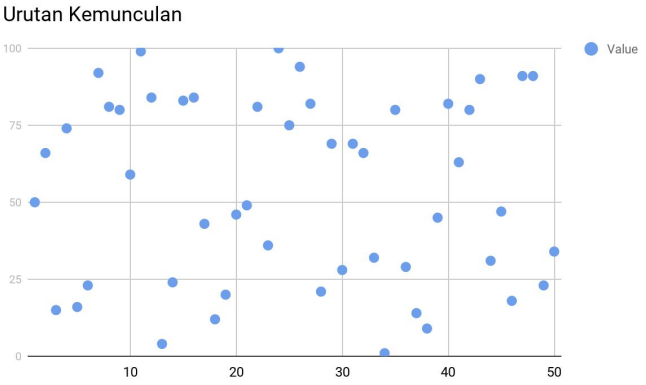
Gambar 5.1.2 Urutan kemunculan

2. **Eksperimen 2**
   N : 50
   S : sebentar lagi libur
   R : 101

   Hasil eksperimen



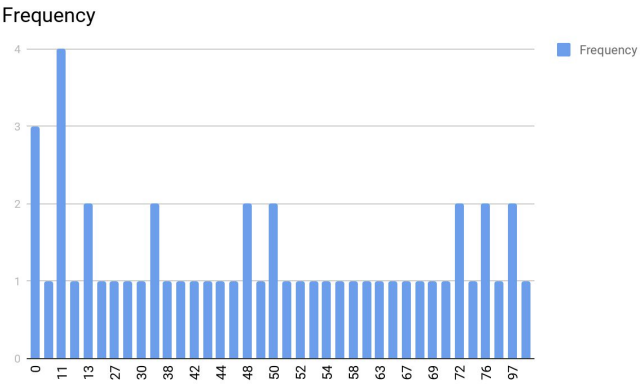
Gambar 5.2.1 Frekuensi kemunculan



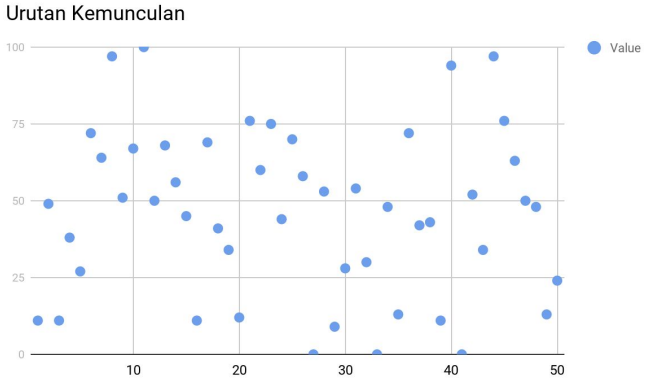
Gambar 5.2.2 Urutan kemunculan

3. **Eksperimen 3**
   N : 50
   S : sebentar lagi lebaran
   R : 101

   Hasil eksperimen



Gambar 5.3.1 Frekuensi kemunculan



Gambar 5.3.2 Urutan kemunculan

Dari ketiga hasil eksperimen yang telah dilakukan dapat dilihat bahwa meskipun nilai *seed* yang digunakan hanya berbeda sebanyak dua karakter saja, tapi rangkaian bilangan acak semu yang dihasilkan sangat berbeda. Hal tersebut dapat dilihat dari diagram yang menggambarkan urutan kemunculan tiap bilangan acak yang dihasilkan.

Dari ketiga eksperimen, dengan nilai N yang masih kecil kita dapat melihat bahwa persebaran bilangan acak yang dihasilkan dari algoritma ini tidak memiliki pola. Selain itu kita juga belum melihat adanya perulangan nilai yang dihasilkan.

VI. Perbandingan dengan Menggunakan Library Random Bahasa Python

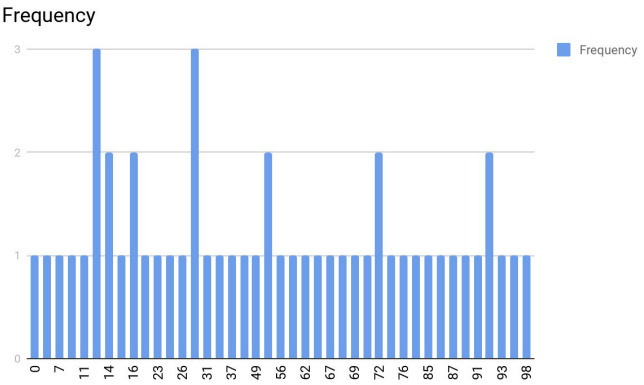
Untuk dapat melihat apakah algoritma pembangkitan bilangan acak yang digunakan dalam makalah ini cukup baik, tentu saja kita perlu membandingkannya dengan algoritma pembangkit bilangan acak yang sudah ada dan sudah digunakan oleh orang banyak.

Pembanding yang akan digunakan dalam makalah ini adalah pembangkit bilangan acak dengan library random pada bahasa pemrograman python.

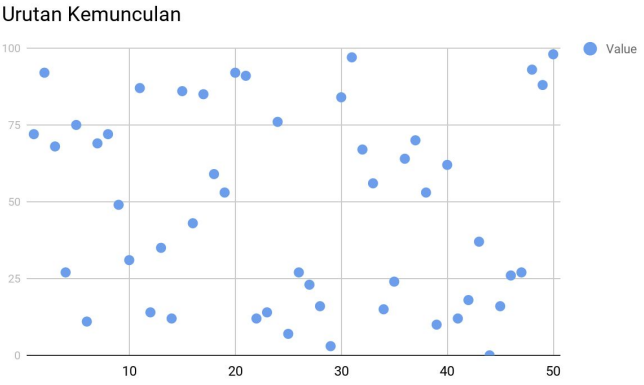
Untuk dapat membandingkan maka akan dilakukan tiga kali eksperimen dengan nilai parameter yang sama dengan eksperimen yang telah dilakukan sebelumnya.

1. **Eksperimen 1**
   N : 50
   S : sebentar lagi liburan
   R : 101

   Hasil eksperimen



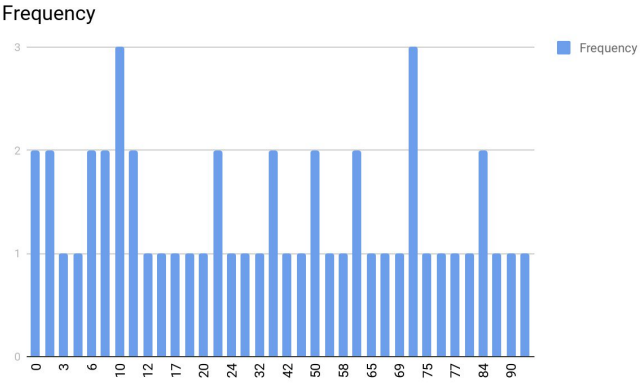
Gambar 6.1.1 Frekuensi kemunculan



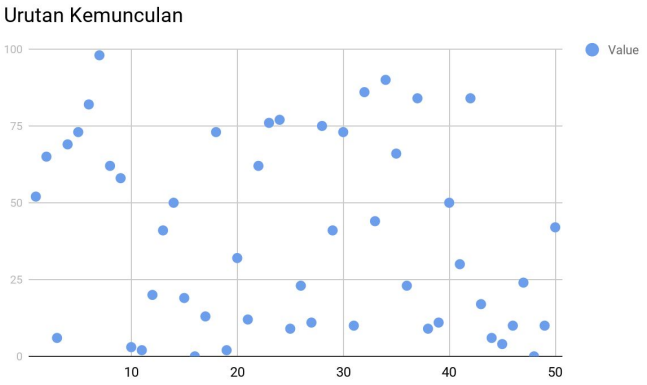
Gambar 6.1.2 Urutan kemunculan

2. **Eksperimen 2**
   N : 50
   S : sebentar lagi libur
   R : 101

   Hasil eksperimen



Gambar 6.2.1 Frekuensi kemunculan

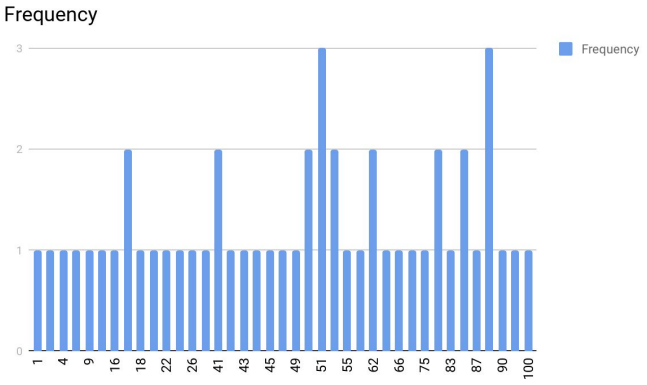


Gambar 6.2.2 Urutan kemunculan

3. Eksperimen 3

N : 50
S : sebentar lagi lebaran
R : 101

Hasil eksperimen

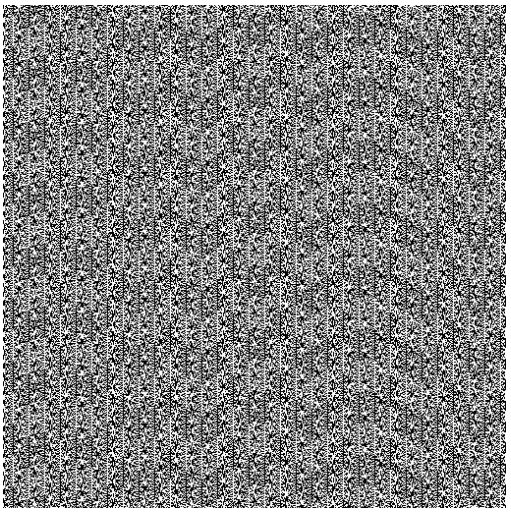


Gambar 6.3.1 Frekuensi kemunculan

VII. Perbandingan pola kemunculan yang dihasilkan

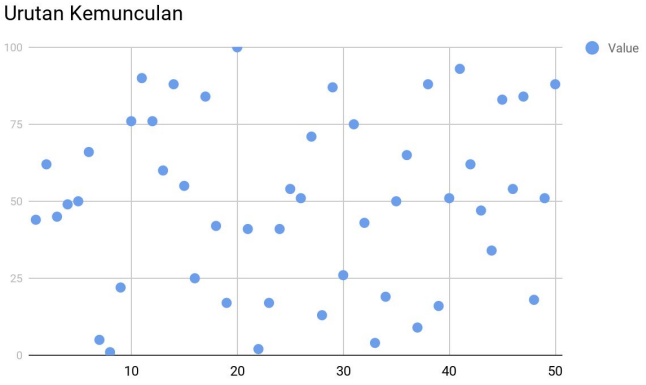
Untuk nilai N yang kecil kita dapat berharap bahwa pola dari pembangkitan nilai acak tidak terlihat dan perulangan nilai bilangan acak juga tidak terlihat. Namun berikut akan ditampilkan hasil pembangkitan bilangan acak untuk nilai N yang besar.

Contoh visualisasi himpunan bilangan acak dimana kita dapat melihat pola yang berulang pada pembangkitannya adalah seperti di bawah ini.



Gambar 7. Visualisasi data dengan pola berulang

Di bawah ini adalah beberapa eksperimen dengan nilai N yang besar

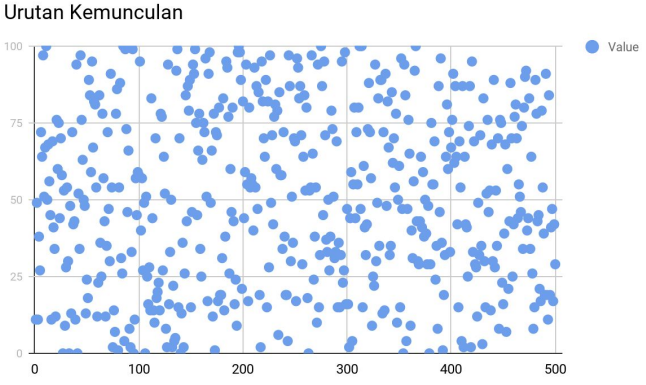


Gambar 6.3.2 Urutan kemunculan

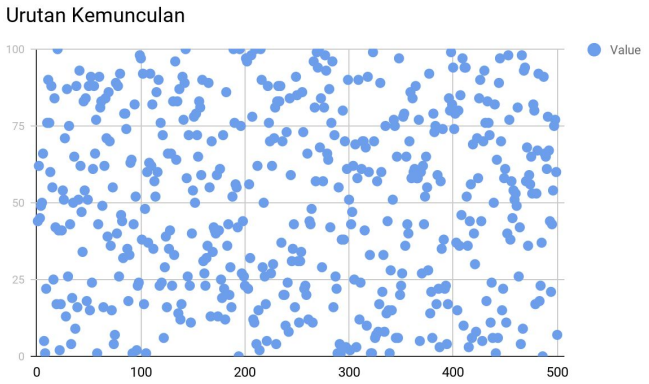
1. Eksperimen 1

N : 500
S : sebentar lagi lebaran
R : 101

Hasil eksperimen



Gambar 7.1.1 Urutan kemunculan bilangan acak dengan algoritma fungsi hash



Gambar 7.1.2 Urutan kemunculan bilangan acak dengan *library* random python

2. Eksperimen 2

N : 5000
S : sebentar lagi lebaran
R : 101

Hasil eksperimen

Gambar 7.2.1 Urutan kemunculan bilangan acak dengan algoritma fungsi hash

Gambar 7.2.2 Urutan kemunculan bilangan acak dengan *library* random python

Dari hasil eksperimen pertama dan kedua, baik hasil pembangkitan bilangan acak dengan penggunaan algoritma fungsi hash maupun hasil pembangkitan bilangan acak dengan penggunaan library random python, keduanya memberikan hasil yang acak.

Dengan nilai N = 500 maupun N=5000, pada keempat diagram di atas kita tidak melihat adanya pola dari urutan kemunculan bilangan acak yang dihasilkan. Selain itu kita juga belum melihat adanya perulangan nilai bilangan yang dihasilkan.

VIII. Kesimpulan

Setelah melakukan beberapa kali eksperimen dan juga melakukan perbandingan dengan algoritma atau *library* pembangkitan bilangan acak yang sudah dikenal dan banyak digunakan orang, dapat dilihat bahwa algoritma pembangkitan bilangan acak yang digunakan dalam makalah ini adalah algoritma yang baik.

Dari hasil eksperimen yang telah dilakukan dapat terlihat bahwa rangkaian nilai bilangan acak yang dihasilkan relatif berbeda meskipun untuk nilai *seed* yang relatif sama. Selain itu urutan kemunculan bilangan acak yang dihasilkan pun tidak memiliki pola yang dapat membedakan dan belum menghasilkan perulangan nilai meskipun saat digunakan untuk menghasilkan banyak bilangan acak yang cukup besar.

IX. Ucapan terima kasih

Syukur Alhamdulillah penulis panjatkan kepada Allah SWT karena berkat rahmat dan karunianya makalah ini dapat selesai dengan baik. Tak lupa juga penulis ingin menyampaikan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, MT. selaku dosen mata kuliah IF4020 Kriptografi yang telah membagikan ilmunya kepada penulis. Selain itu, penulis juga ingin menyampaikan terima kasih kepada kedua orang tua, keluarga, juga sahabat yang selalu memberi dukungan kepada penulis

References

[1] Science and Cyber Security Thomas W. Edgar, David O. Manz, in Research Methods for Cyber Security, 2017
[2] https://www.techopedia.com/definition/19744/hash-function Diakses pada 19 April 2019
[3] https://whatis.techtarget.com/definition/random-numbers Diakses pada 9 mei 2019
[4] https://www.howtogeek.com/183051/htg-explains-how-computers-generate-random-numbers/ Diakses pada 9 mei 2019
[5] https://blog.cloudflare.com/why-randomness-matters/ Diakses pada 9 mei 2019
[6] https://www.techopedia.com/definition/9091/random-number-generator-rng Diakses pada 9 mei 2019
[7] https://en-betfair.custhelp.com/app/answers/detail/a_id/158/~/what-are-random-number-generators%2C-and-how-do-they-work%3F Diakses pada 9 mei 2019
[8] https://www.random.org/randomness/ Diakses pada 9 mei 2019
[9] https://www.random.org/analysis/

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, 10 Mei 2019

Hani'ah Wafa
13516053