Perbandingan Distribusi Keluaran Berbagai Metode Pembangkitan Bilangan Acak Menggunakan Dua Kakas Pembangkit Bilangan Acak C++: rand dan mt19937

Jauhar Wibisono – 13519160 (*Author*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): jauhar.wibisono@gmail.com

***Abstrak*—Pembangkit bilangan acak adalah mesin yang membangkitkan deretan bilangan yang tidak dapat diprediksi nilainya. Salah satu sifat pembangkit bilangan acak yang baik adalah memiliki distribusi keluaran yang seragam. Banyak bahasa pemrograman yang menyediakan kakas pembangkit bilangan acak, termasuk C++ dengan kakas rand dan mt19937. Makalah ini membahas tentang empat metode pembangkitan bilangan acak dengan kakas rand dan mt19937 C++ beserta dengan perbandingan distribusi keluarannya. Didapatkan tabel-tabel dan gambar-gambar yang menunjukkan distribusi keluaran masing-masing metode. Disimpulkan bahwa metode dengan mt19937 tidak lebih buruk dari metode dengan rand dan salah satu metode dengan rand memiliki distribusi sebaik metode dengan mt19937.**

Kata kunci—pembangkit bilangan acak, rand C++, mt19937 C++, distribusi, distribusi seragam

I. PENDAHULUAN

Bilangan acak adalah bilangan yang tidak dapat diprediksi nilainya. Mesin, fungsi, atau prosedur yang membangkitkan bilangan acak biasa disebut pembangkit bilangan acak atau *random number generator*.

Bilangan acak digunakan di berbagai bidang, seperti matematika, fisika, ilmu komputer, rekayasa gim dan kriptografi. Di matematika dan fisika, bilangan acak digunakan untuk mensimulasikan variabel acak dengan metode Monte Carlo. Di ilmu komputer, bilangan acak digunakan dalam berbagai algoritme acak atau *randomized algorithm*. Di rekayasa gim, bilangan acak digunakan untuk mensimulasikan perilaku benda-benda di dunia gim. Di kriptografi, bilangan acak digunakan untuk membangkitkan parameter kunci pada algoritma kunci publik. Terlihat bahwa bilangan acak merupakan topik vital yang mendukung proses dalam berbagai bidang.

Sebagian besar bahasa pemrograman yang sering digunakan sekarang, termasuk C++, memiliki kakas pembangkit bilangan acak. Dalam bahasa C++ terdapat dua kakas pembangkit bilangan acak yang biasa digunakan, yaitu rand dan mt19937 (dan versi 64 bitnya – mt19937_64). Kakas rand diwariskan ke bahasa C++ dari bahasa C, sedangkan mt19937 baru diperkenalkan pada bahasa C++11. Hal tersebut, dipadukan dengan argumen bahwa nama rand lebih intuitif daripada mt19937, membuat kakas rand banyak digunakan. Namun kakas rand menjebak banyak penggunanya karena memiliki distribusi yang tidak merata pada kasus tertentu.

Untuk mengatasi kelemahan tersebut, berbagai metode lain untuk membangkitkan bilangan acak di C++ telah dilakukan. Makalah ini membahas tiga dari berbagai metode lain tersebut beserta perbandingan distribusi keluarannya dengan kakas rand mentah.

II. LANDASAN TEORI

A. Bilangan Acak dan Pembangkit Bilangan Acak

Bilangan acak adalah bilangan yang tidak dapat diprediksi nilainya. Mesin, fungsi, atau prosedur yang membangkitkan bilangan acak biasa disebut pembangkit bilangan acak atau *random number generator*. Meskipun secara ideal keluaran pembangkit bilangan acak tidak dapat diprediksi, namun pembangkit bilangan acak yang biasa digunakan tidak sempurna. Alasannya adalah deretan bilangan acak yang dibangkitkan oleh pembangkit bilangan acak dapat dibangkitkan kembali dengan mengulang langkah-langkah yang sama. Oleh sebab itu, pembangkit bilangan acak yang biasa digunakan disebut juga pembangkit bilangan acak semu atau *pseudo random number generator*.

Pembangkit bilangan acak yang biasa digunakan bekerja dengan menerima suatu nilai dan menghasilkan deretan bilangan acak berdasarkan nilai tersebut. Nilai yang diterima yang diterima di awal tersebut biasa disebut unpan atau *seed*.

Terdapat beberapa sifat yang dimiliki pembangkit bilangan acak yang baik [1], yaitu sebagai berikut.

1. Distribusi nilai keluaran pembangkit bilangan acak yang baik seragam. Tidak ada nilai yang muncul jauh lebih sering daripada suatu nilai lainnya.

2. Keluaran pembangkit bilangan acak yang baik tidak dapat diprediksi, baik nilainya secara utuh maupun nilai bit-bitnya. Terdapat berbagai pengujian yang dapat dipakai untuk mengukur kualitas pembangkit bilangan acak [2]. Pembangkit bilangan acak yang baik dapat melewati tes-tes tersebut dengan baik.
3. Pembangkit bilangan acak yang baik membangkitkan bilangan acak secara cepat dan tidak menjadi *bottleneck* di aplikasi yang menggunakannya. Kecepatannya juga tidak bergantung pada banyak iterasi pembangkitan yang telah dilakukan. Dengan kata lain, pembangkitan bilangan ke-n berjalan secepat pembangkitan bilangan pertama.

Sifat pertama pembangkit bilangan acak yang baik merupakan isu yang dibahas dalam makalah ini.

B. Fungsi rand C++

Menurut dokumentasi C++ [3], rand adalah fungsi yang mengembalikan sebuah nilai *pseudo-random* bertipe *int* antara nol dan konstanta RAND_MAX, inklusif. Umpan pembangkitan yang digunakan rand dapat didefinisikan dengan memanggil fungsi srand dengan masukan umpan. Definisi fungsi rand, srand, dan konstanta RAND_MAX berada dalam header <cstdlib>. Implementasi fungsi rand dan nilai RAND_MAX tergantung versi C++ yang digunakan, namun biasanya fungsi rand diimplementasikan dengan *linear congruential generator* (LCG) dan nilai RAND_MAX dijamin lebih dari atau sama dengan 32767.

Beberapa literatur, bahkan dokumentasi C++ sendiri [3], menyebutkan bahwa kualitas keluaran fungsi rand sebagai bilangan acak tidak baik. Fungsi rand biasa diimplementasikan dengan LCG yang terkenal mudah diprediksi. Dalam kasus yang disebutkan dokumentasi C++, bit kecil keluaran fungsi rand hanya berganti dari nol jadi satu dan sebaliknya di setiap pemanggilan.

Beberapa versi C++ (dan C) menangani hal ini dengan menghapus 16 dari 32 bit terkecil nilai keluaran [4]. Penanganan tersebut menjadi alasan nilai minimum RAND_MAX sama dengan 32767 atau 2^{16-1}-1. Namun penanganan tersebut menimbulkan kelemahan baru, yaitu rentang nilai keluaran fungsi rand menjadi kecil, sehingga dapat menipu pengguna yang ingin mendapatkan nilai besar. Contoh masalah yang ditimbulkan kelemahan ini adalah pengacakan *array* berukuran besar (misal 10^6) dengan fungsi rand tidak menghasilkan pengacakan yang baik [4]. Selisih posisi baru dengan posisi lama untuk semua elemen *array* hanya sebesar 2% dari ukuran *array*.

Fungsi rand dan srand digunakan dalam beberapa metode pembangkitan bilangan acak yang dibahas dalam makalah ini.

C. Kelas mt19937 C++

Kelas mt19937 mulai diperkenalkan pada C++11, bersama dengan *standard library header* <random> [5]. Kelas mt19937 adalah pembangkit bilangan acak berbasis algoritme Mersenne Twister [6]. Dalam implementasinya, mt19937 menggunakan bilangan prima Mersenne $2^{19937} – 1$, sehingga bilangan 19937 muncul dalam namanya.

Algoritme Mersenne Twister yang digunakan kelas mt19937 juga banyak digunakan di bahasa pemrograman lain, seperti Python, PHP, Ruby, dan R. Algoritme Mersenne Twister banyak digunakan karena memiliki kualitas yang jauh lebih baik daripada pembangkit bilangan acak 32 bit seperti fungsi rand C/C++. Kualitas algoritme Mersenne Twister ditunjukkan dengan hasil berbagai macam tes, seperti *Diehard tests* dan *TestU01 tests.*

Kelebihan lain kelas mt19937 adalah mt19937 memiliki rentang keluaran yang besar. Menurut dokumentasi C++ mt19937 [5], rentang keluaran mt19937 adalah [0, 2^w), dengan *w* merupakan konstanta yang ada di mesin. Pada komputer yang biasa digunakan, *w* bernilai 32 sehingga rentang keluaran mt19937 adalah [0, 2^{32}), jauh lebih besar daripada rentang keluaran fungsi rand. Pada *header* <random> bahkan terdapat mt19937 versi 64 bit, yaitu mt19937_64 [5], yang memiliki rentang keluaran yang lebih besar, yaitu [0, 2^{64}). Rentang keluaran yang besar ini sangat membantu dalam berbagai keperluan.

Selain memiliki kualitas algoritme yang baik dan rentang keluaran yang besar, mt19937 membangkitkan bilangan acak dengan lebih cepat. Dalam suatu pengujian dengan mesin Codeforces [4], pembangkitan 10^8 bilangan acak dengan mt19937 berjalan selama 389 milidetik, sedangkan dengan rand berjalan selama 1170 milidetik.

Kelas mt19937 digunakan dalam salah satu metode pembangkitan bilangan acak yang dibahas dalam makalah ini.

III. PEMBAHASAN

A. Persiapan Pengamatan

Metode-metode pembangkitan bilangan acak yang dibahas pada bagian ini adalah sebagai berikut. Sebagai catatan, mulai bagian ini, hasil pemanggilan fungsi rand ditulis sebagai rand().

1. rand()
2. rand() * rand()
3. rand() * RAND_MAX + rand()
4. mt19937 (dengan operator ())

Pengamatan dilakukan dengan membangkitkan sederet bilangan acak, membuat deskripsi statistik deretan bilangan acak tersebut, dan melihat diagram batang deretan bilangan acak tersebut. Pembangkitan deretan bilangan acak dilakukan menggunakan C++17, sedangkan pembuatan deskripsi statistik dan diagram batang dilakukan menggunakan Python 3.8 dan kakas Pandas. Mesin yang digunakan dalam pembangkitan bilangan acak adalah prosesor AMD Ryzen 5 4600U dengan sistem operasi Windows 10.

Kode yang digunakan untuk membangkitkan bilangan acak pada metode-metode dengan rand (metode 1 sampai 3) adalah seperti berikut.

```
#include <bits/stdc++.h>
using namespace std;


const int N_SEED = 10, N_REP = 1e5;
```

```
const int MOD = 100;


int get_rand(){
  // letakkan metode pembangkitan bil. acak di sini
}


int main(){
 for (int i=0; i<N_SEED; i++){
  int t = chrono::steady_clock::now().time_since_epoch().count();
  srand(t);
  for (int j=0; j<N_REP; j++){
   cout << get_rand() % MOD << '\n';
  }
 }
 return 0;
}
```

Gambar 1. Kode Pembangkitan Bilangan Acak dengan rand

Pada kode tersebut, terlihat bahwa umpan pembangkit bilangan acak dibangkitkan berkali-kali dengan jam presisi tinggi. Hal ini dilakukan untuk mengurangi kemungkinan mendapatkan umpan bias. Selain itu, pada kode tersebut terdapat konstanta MOD yang nilainya akan diubah-ubah untuk mengamati kualitas keluaran metode-metode pembangkitan bilangan acak dalam berbagai rentang.

Kode yang digunakan untuk membangkitkan bilangan acak pada metode dengan mt19937 (metode 4) adalah seperti berikut.

```
#include <bits/stdc++.h>
using namespace std;


const int N_SEED = 10, N_REP = 1e5;
const int MOD = 100;


int main(){
 for (int i=0; i<N_SEED; i++){
  int t = chrono::steady_clock::now().time_since_epoch().count();
  mt19937 rng(t);
  for (int j=0; j<N_REP; j++){
   cout << rng() % MOD << '\n';
  }
 }
 return 0;
}
```

Gambar 2. Kode Pembangkitan Bilangan Acak dengan mt19937

Kode tersebut sebagian besar sama dengan kode untuk membangkitkan bilangan acak dengan rand, perbedaannya hanya pada bagian pemberian umpan dan pembangkitan bilangan acak.

Sebagai peringatan, diagram batang yang disajikan di bawah ini tidak sempurna. Secara khusus, diagram batang untuk MOD=1000 tidak disajikan karena mengalami *bug* berat. Untuk diagram batang yang lain, karena nilai MOD yang digunakan besar, banyak kategori yang terbentuk di setiap diagram batang. Akibatnya, lebar tiap batang menjadi sangat kecil dan kurang tergambar di diagram batang. Meskipun demikian, diagram-diagram batang di bawah tetap dapat menunjukkan berbagai hal. Oleh sebab itu, diagram-diagram batang tersebut tetap ditunjukkan.

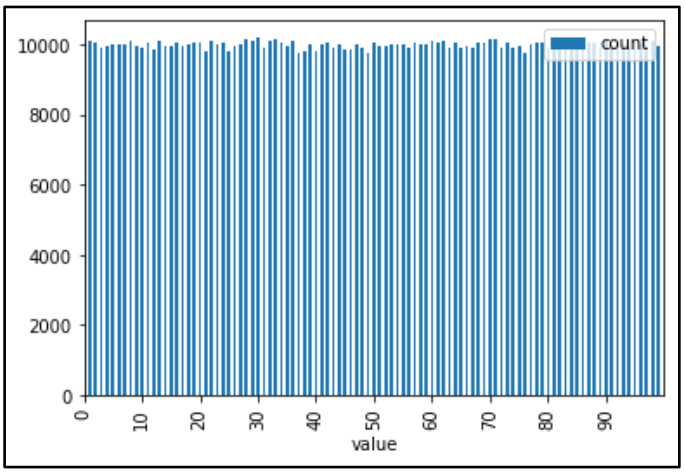
B. Pembangkitan Bilangan Acak Menggunakan rand()

Metode ini membangkitkan bilangan acak hanya dengan memanggil fungsi rand. Seperti yang telah dijelaskan di landasan teori, metode ini terkenal menjebak penggunanya karena memiliki rentang yang kecil.

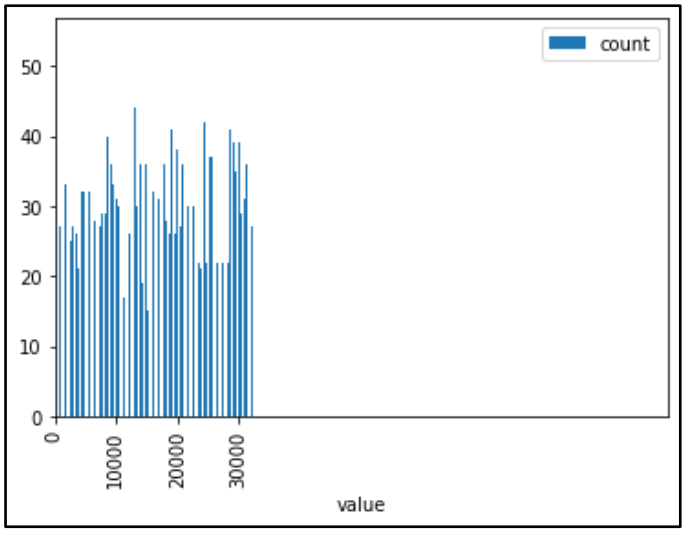
Deskripsi statistik dan diagram batang hasil pembangkitan bilangan acak dengan metode ini adalah sebagai berikut.

MOD = 100	
Rata-rata keluaran	49,48
Frekuensi minimum	9750
Frekuensi maksimum	10212
Ukuran rentang frekuensi	462
MOD = 1000	
Rata-rata keluaran	496,809
Frekuensi minimum	895
Frekuensi maksimum	1114
Ukuran rentang frekuensi	219
MOD = 100000	
Rata-rata keluaran	16382,535
Frekuensi minimum	0
Frekuensi maksimum	54
Ukuran rentang frekuensi	54

Tabel 1. Deskripsi Statistika Metode rand()



Gambar 3. Diagram Batang Metode rand() dengan MOD = 100



Gambar 4. Diagram Batang Metode rand() dengan MOD = 100000

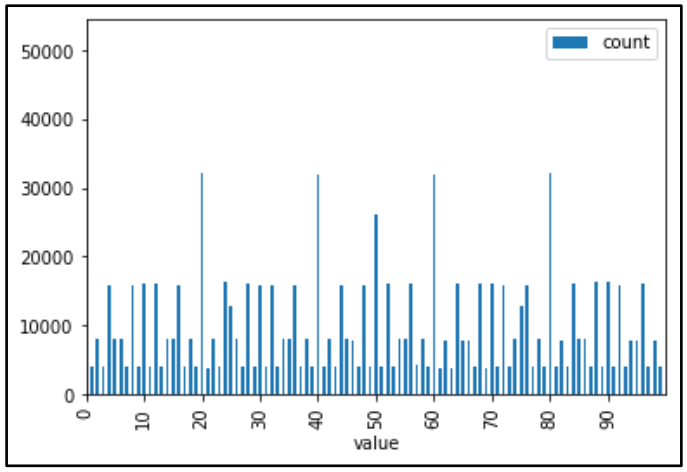
*C. Pembangkitan Bilangan Acak Menggunakan rand()*rand()*

Metode ini membangkitkan bilangan dengan memanggil fungsi rand dua kali, lalu mengalikan hasil keduanya. Dengan mengalikan dua hasil fungsi rand, didapatkan rentang keluaran sekitar [0, 2^{31}) yang mendekati kapasitas penuh tipe data *int* pada C++. Dengan demikian, masalah rentang nilai keluaran fungsi rand kecil terselesaikan.

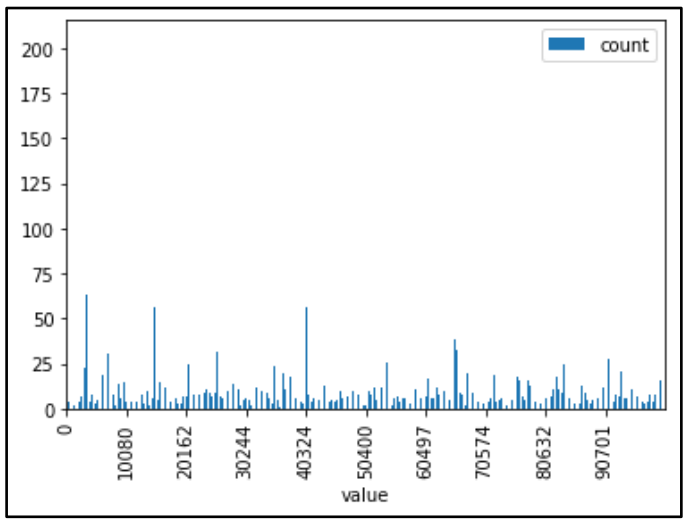
Deskripsi statistik dan diagram batang hasil pembangkitan bilangan acak dengan metode ini adalah sebagai berikut.

MOD = 100	
Rata-rata keluaran	47,37
Frekuensi minimum	3852
Frekuensi maksimum	52025
Ukuran rentang frekuensi	48173
MOD = 1000	
Rata-rata keluaran	495,377
Frekuensi minimum	334
Frekuensi maksimum	8466
Ukuran rentang frekuensi	8122
MOD = 100000	
Rata-rata keluaran	49937,925
Frekuensi minimum	0
Frekuensi maksimum	205
Ukuran rentang frekuensi	205

Tabel 2. Deskripsi Statistika Metode rand()*rand()



Gambar 5. Diagram Batang Metode rand()*rand() dengan MOD = 100



Gambar 6. Diagram Batang Metode rand()*rand() dengan MOD = 100000

*D. Pembangkitan Bilangan Acak Menggunakan rand() * RAND_MAX + rand()*

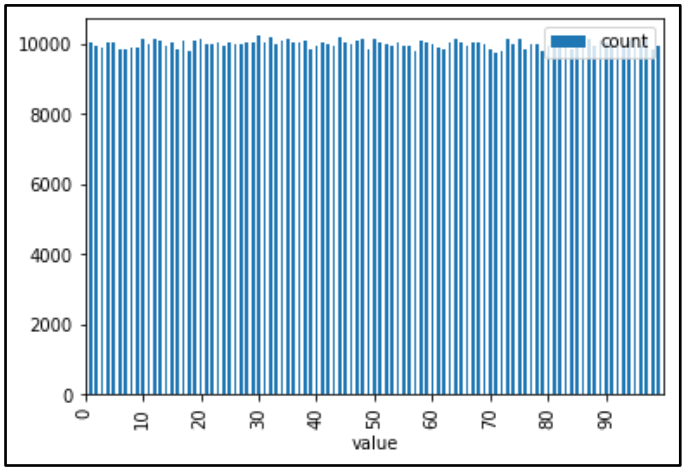
Ide dari metode ini adalah membangkitkan bit-bit besar dan bit-bit kecil secara terpisah. Pada ekspresi rand() * RAND_MAX + rand(), suku rand() * RAND_MAX menjadi bit-bit besar, sedangkan suku rand() menjadi bit-bit kecil. Masing-masing suku memiliki maksimal 16 bit, sehingga rentang keluaran metode ini adalah sekitar [0, 2^{31}). Terlihat bahwa metode ini juga menyelesaikan masalah rentang nilai keluaran fungsi rand yang kecil.

Deskripsi statistik dan diagram batang hasil pembangkitan bilangan acak dengan metode ini adalah sebagai berikut.

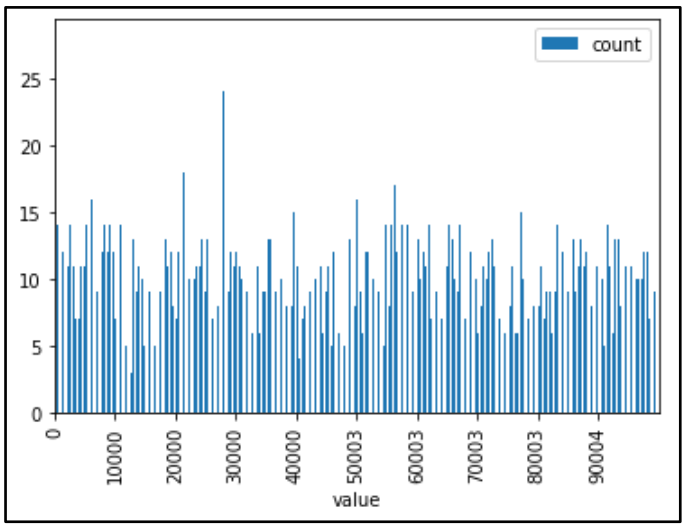
MOD = 100	
Rata-rata keluaran	49,49
Frekuensi minimum	9732
Frekuensi maksimum	10238

Ukuran rentang frekuensi	506
MOD = 1000	
Rata-rata keluaran	499,768
Frekuensi minimum	895
Frekuensi maksimum	1104
Ukuran rentang frekuensi	209
MOD = 100000	
Rata-rata keluaran	50029,868
Frekuensi minimum	0
Frekuensi maksimum	28
Ukuran rentang frekuensi	28

Tabel 3. Deskripsi Statistika Metode rand()*RAND_MAX+rand()



Gambar 7. Diagram Batang Metode rand()*RAND_MAX+rand() dengan MOD = 100



Gambar 8. Diagram Batang Metode rand()*RAND_MAX+rand() dengan MOD = 100000

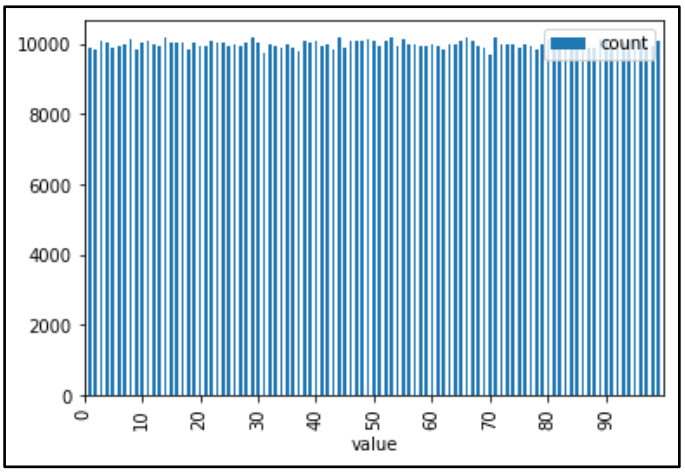
E. Pembangkitan Bilangan Acak Menggunakan mt19937

Metode ini membangkitkan bilangan acak hanya dengan memanggil operator () mt19937. Seperti yang telah dijelaskan di landasan teori, metode ini bagus dan banyak dipakai.

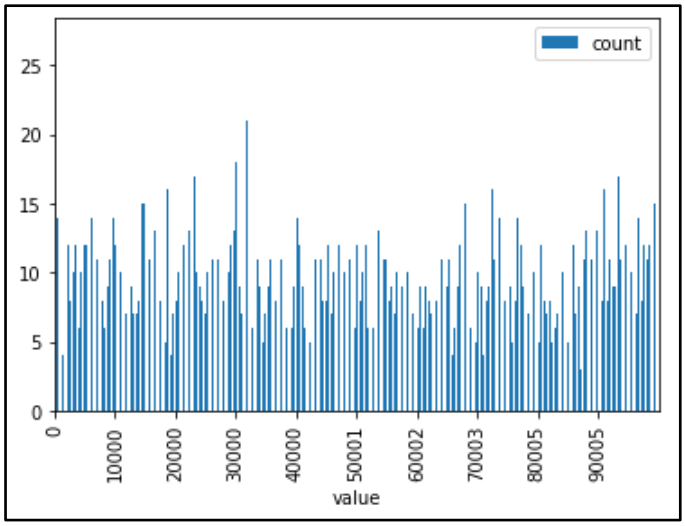
Deskripsi statistik dan diagram batang hasil pembangkitan bilangan acak dengan metode ini adalah sebagai berikut.

MOD = 100	
Rata-rata keluaran	49,51
Frekuensi minimum	9676
Frekuensi maksimum	10185
Ukuran rentang frekuensi	509
MOD = 1000	
Rata-rata keluaran	499,485
Frekuensi minimum	905
Frekuensi maksimum	1101
Ukuran rentang frekuensi	196
MOD = 100000	
Rata-rata keluaran	49984,872
Frekuensi minimum	0
Frekuensi maksimum	27
Ukuran rentang frekuensi	27

Tabel 4. Deskripsi Statistika Metode mt19937



Gambar 9. Diagram Batang Metode mt19937 dengan MOD = 100



Gambar 9. Diagram Batang Metode mt19937 dengan MOD = 100000

F. Perbandingan Distribusi Keluaran

1) Metode Pertama (rand())

Untuk rentang keluaran yang kecil, yaitu hingga 100 dan 1000, distribusi keluaran metode pertama (rand()) cukup baik. Pada Tabel 1, terlihat bahwa rata-rata keluaran cukup dekat dengan rata-rata rentang. Selain itu, terlihat juga bahwa ukuran rentang frekuensi cukup kecil, tidak kalah dengan metode-metode lain. Diagram batang pada Gambar 3 juga mendukung kebaikan tersebut.

Namun metode pertama tidak bekerja dengan baik pada rentang yang besar, yaitu hingga 100000. Pada Tabel 1, terlihat bahwa rata-rata keluaran metode pertama untuk MOD = 100000 hanya 16382,535. Nilai tersebut sangat jauh dari rata-rata rentang yang bernilai sekitar 50000. Selain itu, pada Gambar 4, terlihat bahwa diagram batang kosong untuk nilai sekitar 32000 dan seterusnya. Alasan kinerja yang tidak baik ini sesuai dengan yang dibahas pada landasan teori, yaitu nilai keluaran maksimum dibatasi oleh RAND_MAX yang berukuran sekitar 32000.

Dapat disimpulkan bahwa apabila rentang nilai yang diinginkan tidak besar dan diperlukan distribusi keluaran yang relatif seragam, metode pertama yaitu pembangkitan bilangan acak dengan rand() dapat digunakan.

*2) Metode Kedua (rand()*rand())*

Metode kedua menangani masalah yang dialami metode pertama, yaitu rentang nilai keluaran yang kecil. Pada Gambar 6 terlihat bahwa bagian kanan diagram batang terisi, tidak seperti pada Gambar 5.

Kekurangan yang dimiliki metode kedua adalah distribusi nilai yang buruk dibandingkan dengan metode lainnya. Pada rentang yang kecil yaitu hingga 100 terlihat bahwa rata-rata keluaran metode kedua relatif jauh dari rata-rata rentang. Selain itu, terlihat juga bahwa ukuran rentang frekuensi metode kedua untuk semua rentang relatif besar. Sebagai contoh, pada rentang hingga 1000 (MOD = 1000), ukuran rentang frekuensi metode kedua mencapai 8122, sedangkan pada metode lain hanya berkisar antara 196 sampai 219 saja.

Dapat disimpulkan bahwa metode kedua sebaiknya tidak digunakan karena memiliki distribusi nilai yang relatif buruk, meskipun rentang nilai yang didapatkan mungkin besar.

*3) Metode Ketiga (rand()*RAND_MAX+rand())*

Metode ketiga juga menangani masalah yang dialami metode pertama, yaitu rentang nilai keluaran yang kecil. Pada Gambar 8 terlihat bahwa bagian kanan diagram kanan terisi seperti pada Gambar 6 dan tidak seperti pada Gambar 5. Pada pengamatan ini juga terlihat bahwa metode ketiga tidak memiliki kekurangan. Pada Tabel 3, terlihat bahwa rata-rata keluaran metode ketiga untuk semua rentang cukup dekat dengan rata-rata rentang. Selain itu, terlihat juga bahwa ukuran rentang frekuensi untuk semua rentang cukup kecil, tidak kalah dengan metode-metode lain.

Dapat disimpulkan bahwa apabila diperlukan distribusi keluaran yang relatif seragam, metode ketiga dapat dipakai, bahkan untuk berbagai rentang keluaran. Metode ketiga juga merupakan metode terbaik yang menggunakan kakas/fungsi rand yang dibahas di makalah ini.

4) Metode Keempat (mt19937)

Metode keempat bekerja dengan baik untuk semua rentang. Pada Gambar 9 dan Gambar 10, terlihat bahwa diagram batang cukup penuh dan tidak memiliki bagian kosong. Selain itu, pada Tabel 4, terlihat bahwa rata-rata keluaran cukup dekat dengan rata-rata rentang dan ukuran rentang frekuensi cukup kecil untuk semua rentang. Kualitas metode keempat yang menggunakan kakas mt19937 ini sesuai dengan yang dijelaskan pada landasan teori.

Dapat disimpulkan bahwa apabila diperlukan distribusi keluaran yang relatif seragam, metode keempat dapat dipakai, bahkan untuk berbagai rentang keluaran. Selain itu, sesuai yang dijelaskan di landasan teori, metode ini bagus untuk dipakai karena berjalan lebih cepat daripada metode dengan kakas/fungsi rand.

IV. KESIMPULAN

Sebagai kesimpulan, didapatkan perbandingan distribusi keluaran empat metode pembangkitan bilangan acak menggunakan kakas rand dan mt19937. Perbandingan distribusi keluaran tersebut dapat dilihat pada Tabel 1 hingga Tabel 4 dan Gambar 3 hingga Gambar 10. Didapatkan juga bahwa metode pertama dapat digunakan untuk rentang keluaran kecil, metode kedua sebaiknya tidak digunakan, dan metode ketiga dan keempat dapat digunakan untuk berbagai rentang keluaran (khususnya yang digunakan pada makalah ini).

Makalah ini dapat dikembangkan dengan mengamati distribusi keluaran pada rentang yang lebih besar. Pada makalah ini hal tersebut tidak dapat dilakukan karena keterbatasan kemampuan diagram batang dalam menyajikan informasi. Selain itu, makalah ini dapat dikembangkan dengan mengamati distribusi keluaran metode pembangkitan bilangan acak lain atau pembangkit bilangan acak dalam bahasa lain.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa,
2. orang tua penulis,
3. Bapak Rinaldi Munir sebagai dosen pengampu mata kuliah Kriptografi IF4020 semester Ganjil tahun ajaran 2021/2022,
4. asisten mata kuliah Kriptografi IF4020,
5. teman-teman penulis, dan
6. pihak-pihak lain

yang telah mendukung penulis selama pembelajaran dan proses penulisan makalah ini.

REFERENSI

[1] William, A. et al. *Design of Practical and Provably Good Random Number Generators*. https://doi.org/10.1006/jagm.1998.0952, November 1998.

[2] Unitychain, *Provable Randomness: How to Test RNGs*. https://medium.com/unitychain/provable-randomness-how-to-test-rngs-55ac6726c5a3. 17 September 2019. Diakses 19 Desember 2021.

[3] C++.*std::rand*. https://en.cppreference.com/w/cpp/numeric/random/rand. Diakses 20 Desember 2021.

[4] Wu, Neal. *Don't use rand(): a guide to random number generators in C++*. https://codeforces.com/blog/entry/61587. Diakses 20 Desember 2021.

[5] C++. *std::mersenne_twister_engine*. https://en.cppreference.com/w/cpp/numeric/random/mersenne_twister_engine. Diakses 20 Desember 2021.

[6] Savard, John. *The Mersenne Twister*. http://www.quadibloc.com/crypto/co4814.htm. Diakses 20 Desember 2021.

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.

Yogyakarta, 20 Desember 2021

Jauhar Wibisono, 13519160