

Википедия

Линейный конгруэнтный метод

Материал из Википедии — свободной энциклопедии

**Линейный конгруэнтный метод** — один из методов генерации псевдослучайных чисел. Применяется в простых случаях и не обладает криптографической стойкостью. Входит в стандартные библиотеки различных компиляторов.

Содержание

Описание

Свойства

Часто используемые параметры

Возможность использования в криптографии

См. также

Примечания

Литература

Ссылки

Описание

Линейный конгруэнтный метод был предложен Д. Г. Лемером в 1949 году.<sup>[1]</sup> Суть метода заключается в вычислении последовательности случайных чисел *X<sub>n</sub>*, полагая

$$X_{n+1} = (aX_n + c) \bmod m,$$

где *m* — модуль (натуральное число, относительно которого вычисляет остаток от деления; *m* ≥ 2), *a* — множитель (0 ≤ *a* < *m*), *c* — приращение (0 ≤ *c* < *m*), *X*<sub>0</sub> — начальное значение (0 ≤ *X*<sub>0</sub> < *m*).

Эта последовательность называется *линейной конгруэнтной последовательностью*. Например, для *m* = 10, *X*<sub>0</sub> = *a* = *c* = 7 получим последовательность 7, 6, 9, 0, 7, 6, 9, 0, …<sup>[2]</sup>

Свойства

Линейная конгруэнтная последовательность, определенная числами *m*, *a*, *c* и *X*<sub>0</sub> периодична с периодом, не превышающим *m*. При этом длина периода равна *m* тогда и только тогда, когда<sup>[3]</sup>:

- Числа *c* и *m* взаимно простые;
- c* = *a* − 1 кратно *p* для каждого простого *p*, являющегося делителем *m*;
- c* кратно 4, если *m* кратно 4.

Наличие этого свойства для случая *m* = 2<sup>*e*</sup>, где *e* — число битов в машинном слове, было доказано М. Гринбергом (англ. *M. Greenberg*).<sup>[4]</sup> Наличие этого свойства для общего случая и достаточность условий были доказаны Т. Е. Халлом (англ. *T. E. Hull*) и А. Р. Добеллом (англ. *A. R. Dobell*).<sup>[5]</sup>

Метод генерации линейной конгруэнтной последовательности при  $c = 0$  называют **мультипликативным конгруэнтным методом**, а при  $c \neq 0$  — **смешанным конгруэнтным методом**. При  $c = 0$  генерируемые числа будут иметь меньший период, чем при  $c \neq 0$ , но при определенных условиях можно получить период длиной  $m - 1$ , если  $m$  — простое число. Тот факт, что условие  $c \neq 0$  может приводить к появлению более длинных периодов, был установлен В. Е. Томсоном ([англ. W. T. Thomson](#)) и независимо от него А. Ротенбергом ([англ. A. Rotenberg](#)).<sup>[2]</sup> Чтобы гарантировать максимальность цикла повторения последовательности при  $c = 0$ , необходимо в качестве значения параметра  $m$  выбирать простое число. Самым известным генератором подобного рода является так называемый минимальный стандартный генератор случайных чисел, предложенный Стивеном Парком ([англ. Stephen Park](#)) и Кейтом Миллером ([англ. Keith Miller](#)) в 1988 году. Для него  $a = 16807$ , а  $m = 2147483647$ .<sup>[6][7]</sup>

Наиболее часто практикуемым методом генерации последовательностей псевдослучайных чисел является смешанный конгруэнтный метод.

## Часто используемые параметры

При выборе числа  $m$  необходимо учитывать следующие условия:

- 1) число  $m$  должно быть довольно большим, так как период не может иметь больше  $m$  элементов;
- 2) значение числа  $m$  должно быть таким, чтобы  $(aX_n + c) \bmod m$  вычислялось быстро.

На практике при реализации метода исходя из указанных условий чаще всего выбирают  $m = 2^e$ , где  $e$  — число битов в машинном слове. При этом стоит учитывать, что младшие двоичные разряды сгенерированных таким образом случайных чисел демонстрируют поведение, далёкое от случайного, поэтому рекомендуется использовать только старшие разряды. Подобная ситуация не возникает, когда  $m = w \pm 1$ , где  $w$  — длина машинного слова. В таком случае младшие разряды  $X_n$  ведут себя так же случайно, как и старшие.<sup>[2]</sup> Выбор множителя  $a$  и приращения  $c$  в основном обусловлен необходимостью выполнения условия достижения периода максимальной длины.

### Таблица хороших констант для линейных конгруэнтных генераторов

Все приведенные константы обеспечивают работу генератора с максимальным периодом. Таблица упорядочена по максимальному произведению, которое не вызывает переполнение в слове указанной длины.<sup>[8]</sup>

Переполняется при	a	c	m
2 <sup>20</sup>	106	1283	6075
2 <sup>21</sup>	211	1663	7875
2 <sup>22</sup>	421	1663	7875
2 <sup>23</sup>	430	2531	11979
2 <sup>23</sup>	936	1399	6655
2 <sup>23</sup>	1366	1283	6075
2 <sup>24</sup>	171	11213	53125
2 <sup>24</sup>	859	2531	11979
2 <sup>24</sup>	419	6173	29282
2 <sup>24</sup>	967	3041	14406
2 <sup>25</sup>	141	28411	134456
2 <sup>25</sup>	625	6571	31104
2 <sup>25</sup>	1541	2957	14000
2 <sup>25</sup>	1741	2731	12960
2 <sup>25</sup>	1291	4621	21870
2 <sup>25</sup>	205	29573	139968
2 <sup>26</sup>	421	17117	81000
2 <sup>26</sup>	1255	6173	29282
2 <sup>26</sup>	281	28411	134456
2 <sup>27</sup>	1093	18257	86436
2 <sup>27</sup>	421	54773	259200
2 <sup>27</sup>	1021	24631	116640
2 <sup>28</sup>	1277	24749	117128
2 <sup>28</sup>	2041	25673	121500
2 <sup>29</sup>	2311	25367	120050
2 <sup>29</sup>	1597	51749	244944
2 <sup>29</sup>	2661	36979	175000
2 <sup>29</sup>	4081	25673	121500
2 <sup>29</sup>	3661	30809	145800
2 <sup>30</sup>	3877	29573	139968
2 <sup>30</sup>	3613	45289	214326
2 <sup>30</sup>	1366	150889	714025
2 <sup>31</sup>	8121	28411	134456
2 <sup>31</sup>	4561	51349	243000
2 <sup>31</sup>	7141	54773	259200
2 <sup>32</sup>	9301	49297	233280

$2^{32}$	4096	150889	714025
$2^{33}$	2416	374441	1771875
$2^{34}$	17221	107839	510300
$2^{34}$	36261	66037	312500
$2^{35}$	84589	45989	217728

Печально известен «неудачный» (с точки зрения качества выходной последовательности) алгоритм RANDU, на протяжении многих десятилетий использовавшийся в самых разных компиляторах.

Для улучшения статистических свойств числовой последовательности во многих генераторах псевдослучайных чисел используется только часть битов результата. Например, в стандарте ISO/IEC 9899 на язык Си приведен (но не указан в качестве обязательного) пример функции `rand()`, принудительно отбрасывающей младшие 16 и один старший разряд.

```
#define RAND_MAX 32767

static unsigned long int next = 1;

int rand(void)
{
    next = next * 1103515245 + 12345;
    return (unsigned int)(next/65536) % (RAND_MAX + 1);
}

void srand(unsigned int seed)
{
    next = seed;
}
```

Именно в таком виде функция `rand()` используется в компиляторах Watcom C/C++. Известны числовые параметры иных алгоритмов, применяемых в различных компиляторах и библиотеках.

Source	<i>m</i>	множитель <i>a</i>	слагаемое <i>c</i>	используемые биты
<i>Numerical Recipes</i> <sup>[9]</sup>	2 <sup>32</sup>	1664525	1013904223	
Borland C/C++	2 <sup>32</sup>	22695477	1	bits 30..16 in <i>rand()</i> , 30..0 in <i>lrand()</i>
<i>glibc</i> (used by <i>GCC</i> ) <sup>[10]</sup>	2 <sup>31</sup>	1103515245	12345	bits 30..0
ANSI C: <i>Watcom</i> , <i>Digital Mars</i> , <i>CodeWarrior</i> , <i>IBM VisualAge C/C++</i> <sup>[11]</sup>	2 <sup>31</sup>	1103515245	12345	bits 30..16
C99, C11: Suggestion in the ISO/IEC 9899 <sup>[12]</sup>	2 <sup>32</sup>	1103515245	12345	bits 30..16
Borland Delphi, Virtual Pascal	2 <sup>32</sup>	134775813	1	bits 63..32 of ( <i>seed</i> * <i>L</i> )
Microsoft Visual/Quick C/C++	2 <sup>32</sup>	214013 (343FD <sub>16</sub> )	2531011 (269EC3 <sub>16</sub> )	bits 30..16
Microsoft Visual Basic (6 and earlier) <sup>[13]</sup>	2 <sup>24</sup>	1140671485 (43FD43FD <sub>16</sub> )	12820163 (C39EC3 <sub>16</sub> )	
RtlUniform from <i>Native API</i> <sup>[14]</sup>	2 <sup>31</sup> − 1	2147483629 (7FFFFFFD <sub>16</sub> )	2147483587 (7FFFFFFC3 <sub>16</sub> )	
Apple CarbonLib, C++11's <i>minstd_rand0</i> <sup>[15]</sup>	2 <sup>31</sup> − 1	16807	0	see <i>MINSTD</i>
C++11's <i>minstd_rand</i> <sup>[15]</sup>	2 <sup>31</sup> − 1	48271	0	see <i>MINSTD</i>
MMIX by Donald Knuth	2 <sup>64</sup>	6364136223846793005	1442695040888963407	
Newlib	2 <sup>64</sup>	6364136223846793005	1	bits 63...32
VAX's <b>MTH\$RANDOM</b> , <sup>[16]</sup> old versions of <i>glibc</i>	2 <sup>32</sup>	69069	1	
Java	2 <sup>48</sup>	25214903917	11	bits 47...16
<b>Ранее во многих компиляторах:</b>				
<i>RANDU</i>	2 <sup>31</sup>	65539	0	

## Возможность использования в криптографии

Хотя линейный конгруэнтный метод порождает статистически хорошую псевдослучайную последовательность чисел, он не является криптографически стойким. Генераторы на основе линейного конгруэнтного метода являются предсказуемыми, поэтому их нельзя использовать в криптографии. Впервые генераторы на основе линейного конгруэнтного метода были взломаны Джимом Ридсом (Jim Reeds), а затем Джоан Бояр (Joan Boyar). Ей удалось также вскрыть квадратические и кубические генераторы. Другие исследователи расширили идеи Бояр, разработав способы вскрытия любого полиномиального генератора. Таким образом, была доказана бесполезность генераторов на основе конгруэнтных методов для криптографии. Однако генераторы на основе линейного конгруэнтного метода сохраняют свою полезность для некриптографических приложений, например, для моделирования. Они эффективны и в большинстве используемых эмпирических тестах демонстрируют хорошие статистические характеристики<sup>[8]</sup>.

## См. также

- Генератор псевдослучайных чисел

- Инверсный конгруэнтный метод

## Примечания

1. D. H. Lehmer, Mathematical methods in large-scale computing units, Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery, 1949, Harvard University Press, Cambridge, Mass., 1951, pp. 141—146. MR 0044899 (13,495f)[1] (<http://www.ams.org/journals/bull/1978-84-06/S0002-9904-1978-14532-7/>)
2. *Дональд Кнут*. Том 2. Получисленные методы // Искусство программирования. Указ. соч. — С. 21—37.
3. Кнут Д. Э., Искусство программирования. Том 2. Получисленные методы — Вильямс. 2001. с.21-37
4. M. Greenberger, Method in randomness, Comm. ACM 8 (1965), 177—179.[2] (<http://www.ams.org/journals/bull/1978-84-06/S0002-9904-1978-14532-7/>)
5. T.E. Hull and A.R. Dobell «Random Number Generators»,SIAM Review 4-3(1962),230-254 [3] ([https://dspace.library.uvic.ca:8443/bitstream/handle/1828/3142/Random\\_Number\\_Generators.pdf?sequence](https://dspace.library.uvic.ca:8443/bitstream/handle/1828/3142/Random_Number_Generators.pdf?sequence))
6. "Бакнелл Д. М. Фундаментальные алгоритмы и структуры данных в Delphi. Библиотека программиста. 2002 год. журнал Delphi Informant Magazine. Глава 6.
7. Stephen K. Park and Keith W. Miller (1988). Random Number Generators: Good Ones Are Hard To Find. Communications of the ACM 31 (10): 1192—1201[4] (<http://dl.acm.org/citation.cfm?doid=63039.63042>)
8. *Брюс Шнайер*. Глава 16. // Прикладная криптография.Триумф.2002. Указ. соч. — С. 275.[5] ([http://www.ssl.stu.neva.ru/psw/crypto/appl\\_rus/appl\\_cryp.htm](http://www.ssl.stu.neva.ru/psw/crypto/appl_rus/appl_cryp.htm)) Архивная копия ([http://web.archive.org/web/20140228081623/http://www.ssl.stu.neva.ru/psw/crypto/appl\\_rus/appl\\_cryp.htm](http://web.archive.org/web/20140228081623/http://www.ssl.stu.neva.ru/psw/crypto/appl_rus/appl_cryp.htm)) от 28 февраля 2014 на Wayback Machine
9. Numerical recipes in C. The art of scientific computing. 2-nd edition. - Cambridge University Press, 1992. - 925 pp.
10. The GNU C library's *rand()* in *stdlib.h* uses a simple (single state) linear congruential generator only in case that the state is declared as 8 bytes. If the state is larger (an array), the generator becomes an additive feedback generator and the period increases. See the simplified code (<http://www.mscs.dal.ca/~selinger/random/>) that reproduces the random sequence from this library.
11. A collection of selected pseudorandom number generators with linear structures, K. Entacher, 1997 (<http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.53.3686&rep=rep1&type=pdf>). Дата обращения 16 июня 2012.
12. Last public Committee Draft from April 12, 2011, page 346f (<http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf>) . Дата обращения 21 декабря 2014.
13. How Visual Basic Generates Pseudo-Random Numbers for the RND Function (<http://support.microsoft.com/kb/231847>). *Microsoft Support*. Microsoft. Дата обращения 17 июня 2011.
14. In spite of documentation on MSDN ([http://msdn.microsoft.com/en-us/library/bb432429\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/bb432429(VS.85).aspx)), RtlUniform uses LCG, and not Lehmer's algorithm, implementations before *Windows Vista* are flawed, because the result of multiplication is cut to 32 bits, before modulo is applied
15. ISO/IEC 14882:2011 ([http://www.iso.org/iso/iso\\_catalogue/catalogue\\_tc/catalogue\\_detail.htm?csnumber=50372](http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50372)). ISO (2 September 2011). Дата обращения 3 сентября 2011.
16. GNU Scientific Library: Other random number generators ([http://www.gnu.org/software/gsl/manual/html\\_node/Other-random-number-generators.html](http://www.gnu.org/software/gsl/manual/html_node/Other-random-number-generators.html))

## Литература

- *Дональд Э. Кнут*. Глава 3. Случайные числа // Искусство программирования = The Art of Computer Programming. — 3-е изд. — М.: Вильямс, 2000. — Т. 2. Получисленные алгоритмы. — 832 с. — 7000 экз. — ISBN 5-8459-0081-6 (рус.) ISBN 0-201-89684-2 (англ.).

## Ссылки

- *Л. Бараи*. Алгоритм AKS проверки чисел на простоту и поиск констант генераторов псевдослучайных чисел (<https://web.archive.org/web/20141018205118/http://www.comphys.ru/files/bit-2.pdf>) // Безопасность информационных технологий. — 2005. — № 2. — С. 27-38.
- *Stephen K. Park and Keith W. Miller*. Random Number Generators: Good Ones Are Hard To Find (<http://www.firstpr.com.au/dsp/rand31/p1192-park.pdf>) . — 1988. — № 2. — С. 1192-1201.
- *Бакнелл Д.М.* Фундаментальные алгоритмы и структуры данных в Delphi.Библиотека программиста.[6] (<http://www.books.ru/books/fundamentalnye-algoritmy-i-struktury-dannykh-v-delphi-biblioteka-programmista-414787/>) // Delphi Informant Magazine (<http://www.delphizine.com/>). — 2002.

Источник — [https://ru.wikipedia.org/w/index.php?title=Линейный\\_конгруэнтный\\_метод&oldid=99935911](https://ru.wikipedia.org/w/index.php?title=Линейный_конгруэнтный_метод&oldid=99935911)

**Эта страница в последний раз была отредактирована 21 мая 2019 в 10:05.**

Текст доступен по лицензии Creative Commons Attribution-ShareAlike; в отдельных случаях могут действовать дополнительные условия.

Wikipedia® — зарегистрированный товарный знак некоммерческой организации Wikimedia Foundation, Inc.