Generátory pseudonáhodných čísel I - obecný úvod

Karel Břinda
karel.brinda@fjfi.cvut.cz

Theoretical **I**nformatics **Gr**oup
FJFI ČVUT v Praze

Combinatorics on Words 2011
Fryšava, 17. května 2011

O čem bude řeč

- Motivace, využití generátorů
- Co to je náhodná posloupnost
- Jak náhodnost testovat
- Se kterými generátory se nejčastěji setkáme a na co si dát pozor

Motivace

Kde všude se nám hodí náhodná čísla

- Simulace
- Numerická analýza - např. výpočet čísla π
- Programování - testy efektivity algoritmů, náhodnostní „algoritmy“ (např. genetické alg.), ...
- Rozhodování - losování u veřejných zakázek, ...
- Kryptografie
- Zábava

Konkrétní využití + požadavky na generátor

Způsob využití	**Požadavky na generátor**
Počítačové hry	(žádné :)
Kryptografie - proudové šifry	Kvalita Rychlost Zrekonstruovatelnost
Kryptografie - generování klíčů	Kvalita
Výpočty metodou Monte Carlo	Kvalita Rychlost Zrekonstruovatelnost

Posloupnost náhodných čísel

Různé způsoby chápání

- Intuitivně
- Statisticky
- Přes vyčíslitelnost - Turingův stroj

Typy generátorů

- Generátor náhodných čísel (RNG)
- Generátor pseudonáhodných čísel (PRNG)

Generátory náhodných čísel (RNG)

- Nedeterministický zdroj + funkce na jeho zpracování (destilační funkce)
- Obvykle měření nějaké fyzikální veličiny (např. elektronický šum), pohyb myši uživatele, ...
- Takto získaná posloupnost nelze zrekonstruovat
- Pomalé

Generátory pseudonáhodných čísel (PRNG)

- Algoritmus generující pro daný vstup (takzvaný seed) posloupnost čísel
  - Seed musí být volen také náhodně

$$X_i = f(X_{i-1}, \ldots, X_{i-j})$$

- Takto získaná posloupnost lze se znalostí seedu zrekonstruovat
- Dobré generátory produkují lepší posloupnosti než fyzikální zdroje
- Základní pravidlo: Náhodná čísla nelze generovat náhodně zvolenou metodou. Výpočet musí být podložen teorií.
- Základní omyl: Pokud vezmeme dobrý generátor a trošku ho upravíme, dostaneme stejně tak dobrý nebo lepší

Požadavky na PRNG

- Co nejdelší perioda
- Rovnoměrné rozložení čísel X_i
- Mezi členy $X_i, \ldots, X_{i+j}$ ani $f(X_i), \ldots, f(X_{i+j})$, kde f je nějaká funkce, nesmí být žádný vztah
- Efektivita (rychlost generování a paměťová náročnost)
- Reprodukovatelnost

Testování

- Mnoho různých testů hledajících nenáhodnosti, existují ucelené baterie testů
  - STS (Statistical test suite)
  - DIEHARD

Testy

Frekvenční test

- Monobitový - Zkoumá počet 0 a 1 v celé předané posloupnosti
- V blocích - Zkoumá počet 0 a 1 v blocích o M bitech

Test sérií

- Zkoumá délky podposloupností stejných bitů

Testy

Test hodností binární matice

- Testovanou posloupností se naplní několik matic
- Spočítají se jejich hodnosti

Spektrální test

- Založeno na diskrétní Fourierově transformaci - zkoumají se šířky píků
- Detekuje periodičnosti posloupnosti (opakující se vzory blízko sebe)

Testy

Porovnávací test s překrýváním a bez překrývání

- Cílem testu je odhalit, zda se nevyskytují v posloupnosti příliš často některé předdefinované vzory
- Lze si představit jako posouvání okénka o dané délce po posloupnosti a hledání daného vzoru

Mauerův univerzální statistický test

- Zkoumá, zda není možné testovanou posloupnost zkomprimovat

Testy

Test lineární složitosti

- Testovaná posloupnost bitů se rozdělí postupně do několika stejně dlouhých bloků
- U každého bloku se otestuje, jak složitý LFSR by byl potřeba, aby vygeneroval právě tento blok

Sériový test

- Zkoumá překryvy m-bitových vzorů v rámci testované posloupnosti
- Pro dané m by měly mít všechny m-bitové vzory stejnou pravděpodobnost výskytu

Nejznámější PRNG

- Kongruenční generátory
  - Lineární kongruenční generátor (LCG)
  - Kvadratický kongruenční generátor (QCG)
- Lineární posuvné registry se zpětnou vazbou (LFSR)
- Mersenne twister
- Blum-Blum-Shub (BBSG)

Lineární kongruenční generátor[1]

- Nejrozšířenější typ PRNG
- Základní podoba:

Hlavní vzorec	$X_{n+1} = (aX_n + c) \mod m$
Vztah pro $(n+k)$. člen	$X_{n+k} = (a^k X_n + \frac{a^k - 1}{a-1} c) \mod m$
Konstanty	$0 < \mathbf{m}$ modul $2 \leq \mathbf{a} < m$ násobitel $0 \leq \mathbf{c} < m$ inkrement $0 \leq \mathbf{X_0} < m$ seed

- Minimální délka cyklu v permutaci X_n - perioda generátoru

[1] LCG - Linear congruential generator

Výběr konstant

- Modul m
  - Musí být dostatečně velký (perioda vždy $\leq$)
  - Nesmí být příliš velký (rychlost výpočtu) - z výp. hlediska ideálně $m = 2^{\text{velikost použitého registru na procesoru}}$ (problém - nejnižší bity výsledku jsou „málo náhodné“)
- Násobitel a a inkrement c

Věta

Lineární kongruentní posloupnost definovaná parametry m, a, c a X_0 má periodu délky m právě tehdy, když

- *c a m jsou nesoudělné;*
- *$a - 1$ je násobkem každého prvočísla, které dělí m;*
- *$a - 1$ je násobkem 4, pokud je i m násobkem 4.*

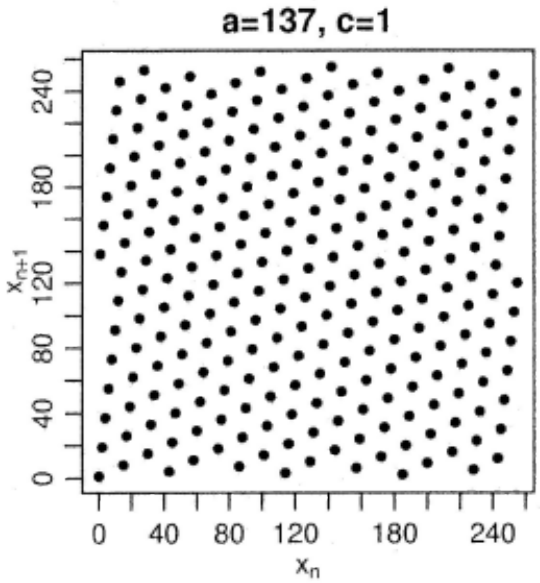
- X_0 libovolně, často se odvozuje z aktuálního systémového času

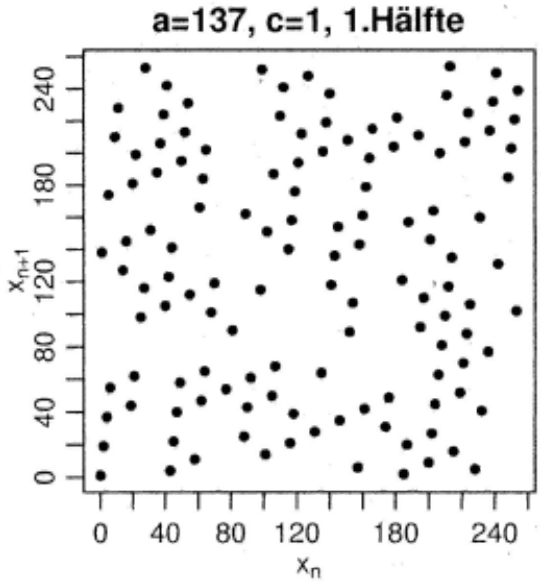
Několik receptů k volbě konstant

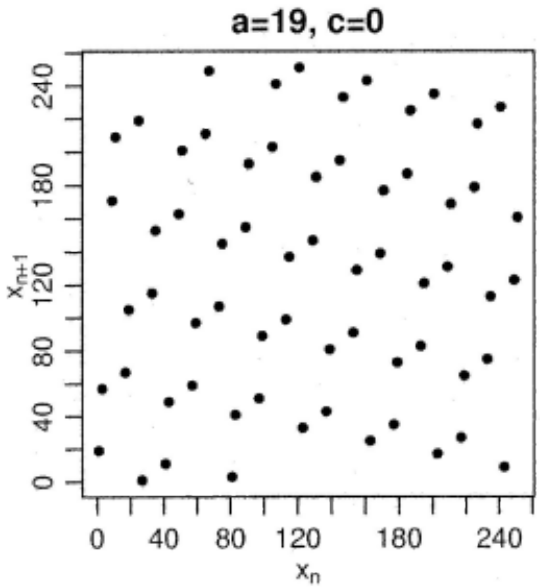
- Je-li $m = 2^n$, najdeme a takové, že $a \mod 8 = 5$
- Je-li $m = 10^n$, najdeme a takové, že $a \mod 200 = 21$
- a hledáme nejlépe v intervalu $0,01m - 0,99m$, ve dvojkovém a desítkovém rozvoji by neměl mít jednoduchý, pravidelný vzorek
- c nesmí být soudělné s m, často se volí $c = 1$ nebo $c = a$

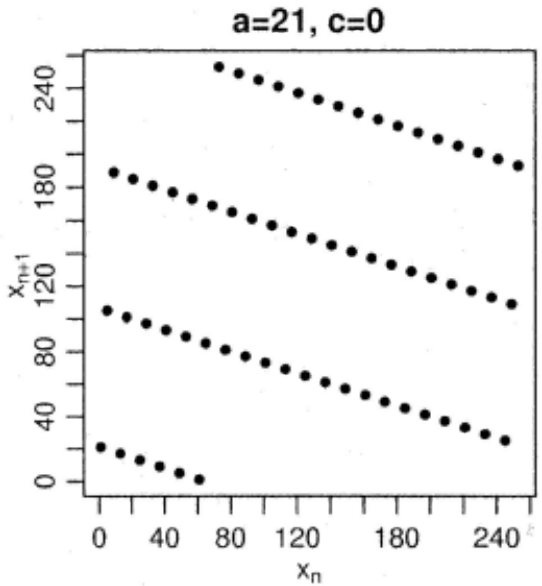
Výhody a nevýhody LCG

Výhody	Rychlý
	Jednoduše naprogramovatelný
Nevýhody	V nějakém n-rozměrném prostoru umisťuje všechny body do několika nadrovin
	Bity nižších řádů nejsou tolik náhodné
	Předvídatelný


a=137, c=1
0
40
80
120
180
240
x_{n+1}
x_n


a=137, c=1, 1.Hälfte
0
40
80
120
180
240
x_{n+1}
x_n


a=19, c=0
0
40
80
120
180
240
x_{n+1}
x_n


a=21, c=0
0
40
80
120
180
240
x_{n+1}
x_n

Implementace LCG

Implementace	**m**	**a**	**c**	**Použité bity**
Borland C/C++	2^{32}	22 695 477	1	30 . . . 16 v *rand()* 30 . . . 0 v *lrand()*
glibc [2]	2^{32}	1 103 515 245	12 345	30 . . . 0
ANSI C [3]	2^{32}	1 103 515 245	12 345	30 . . . 16
Borland Delphi	2^{32}	134 775 813	1	63 . . . 32
Microsoft Visual C++	2^{32}	214 013	2 531 011	30 . . . 16
Java API - Random class	2^{48}	25 214 903 917	11	47 . . . 16

[2] Překladač GCC
[3] Překladače Watcom, Digital Mars, CodeWarrior, IBM VisualAge C/C++

Blum-Blum-Shub[4]

- Konkrétní varianta kvadratického kongruenčního generátoru
- Vzorec

$$X_{n+1} = X_n^2 \mod M,$$

kde $M = pq$ je součin velkých prvočísel. Ideálně

$$p \mod 4 = 3 \text{ a } q \mod 4 = 3$$

- Není vhodný pro simulace metodou Monte Carlo (pomalý), hodí se pro kryptografii

[4] BBSG

Zpožděný Fibonacciho generátor[5]

$$X_n = \bigoplus(X_{n-i_1}, \ldots, X_{n-i_j}) \mod m$$

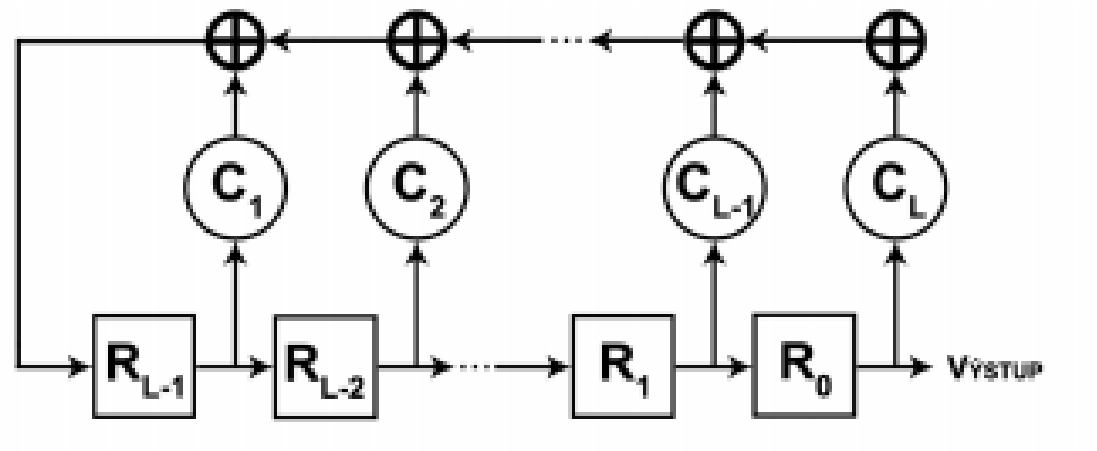
- Binární operace $\bigoplus$ (sčítání, odčítání, násobení, XOR, ...)

[5] LFG - Lagged Fibonacci generator

Mersenne twister

- Jeden z nejlepších generátorů
- Navržený s ohledem na použití v simulacích metodou Monte Carlo
- Není vhodný pro kryptografii (ze znalosti jistého počtu členů lze odvodit pokračování posloupnosti)
- Výstupem posloupnost *uint*32, periodou některé z Mersennových prvočísel
- Na vygenerování seedu se často používá LCG

Lineární posuvný registr se zpětnou vazbou[6]

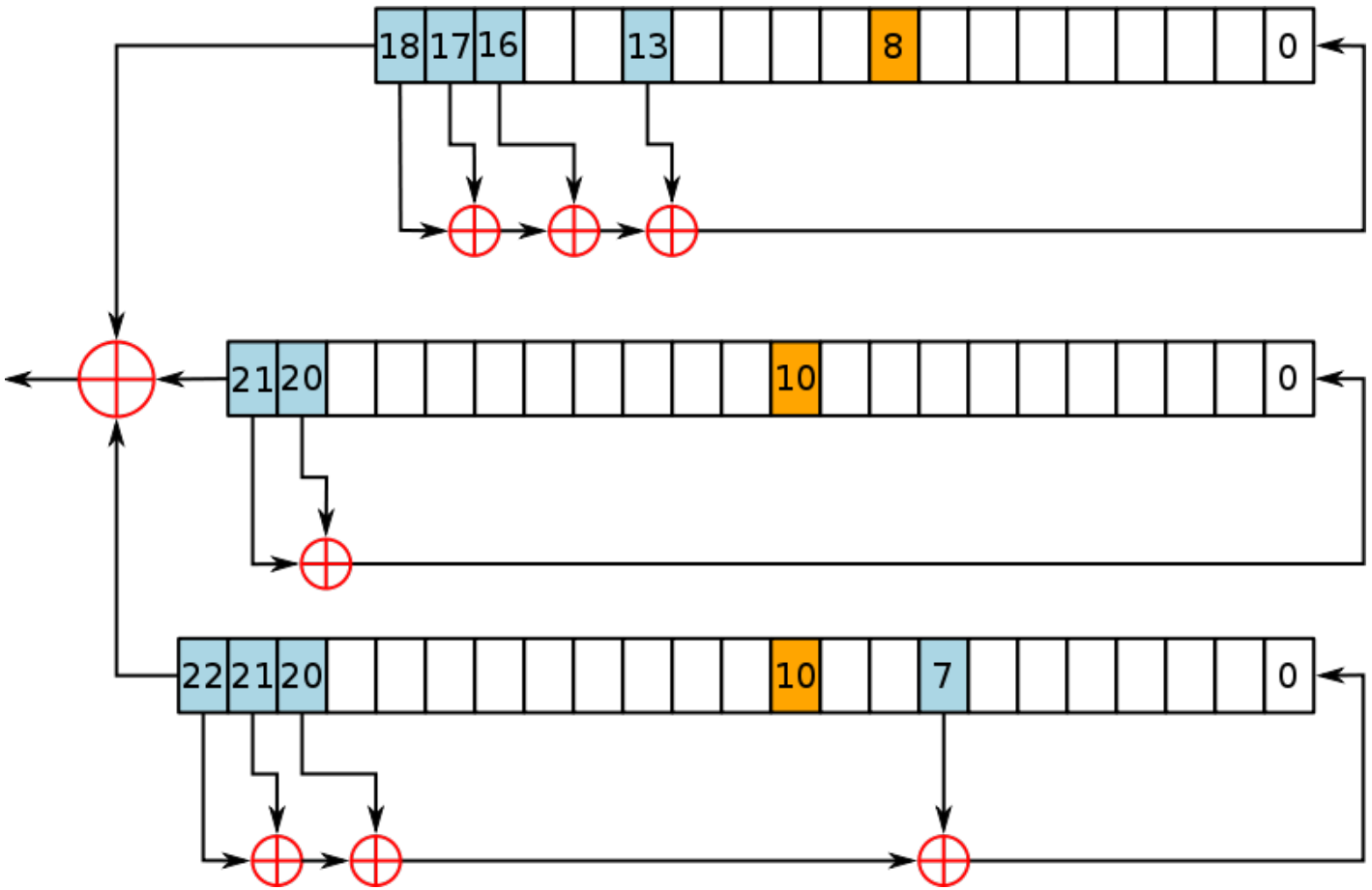


- L registrů, lineární binární operace $\bigoplus$ (nejčastěji XOR), charakteristický polynom
- V jednom taktu: obsah registru R_i se přesune do R_{i-1}, R_0 se předá na výstup; pro registr zpětné vazby R_{L-1} se spočítá nový obsah (pomocí char. polynomu)

[6] LFSR - Linear feedback shift register

A5/1 - šifrování komunikace mobilních telefonů

Blok tří LSFR:



Shrnutí

- Náhodnost
- Testování generátorů
- Nejpoužívanější generátory - lineární kongruenční, Blum-Blum-Shub, Mersenne twister, lineární posuvný se zpětnou vazbou

Reference


D. Knuth: **Umění programování, 2.díl - Seminumerické algoritmy**, Computer Press, 2010.


WIKIPEDIA - The Free Encyclopedia, hesla: A5/1, Linear feedback shift register, Linear congruential generator, Pseudorandom number generator, Blum Blum Shub, Lagged Fibonacci generator, Mersenne twister.


A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo: **A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications**, 2010.


O. Málek: **Generování pseudonáhodných dat založené na použití LFSR** (bakalářská práce), Masarykova univerzita, 2007.


M. Kaňok: **Generátory pseudonáhodných čísiel v metódach Monte Carlo**, Československý časopis pro fyziku, 2008/6.

Děkuji za pozornost.