Slučajni brojevi

Katarina Šupe

Zagreb, 2. lipnja 2020.

Sadržaj

1 Uvod

Slučajni brojevi, tj. nizovi slučajnih brojeva danas pronalaze svoju primjenu u raznim područjima, od kojih je bitno spomenuti kriptografiju, kockarnice, računalne igre, simulacije te brojna znanstvena istraživanja. Slučajni brojevi koriste se u računarskoj znanosti, stoga računalima dajemo zadaću generiranja nekog slučajnog broja. Međutim, znamo da računalo „slijepo" prati zadane upute pa je važno na koji način pristupimo generiranju slučajnih brojeva, da bi se takvi brojevi doimali uistinu slučajnima. U ovom radu pričat ćemo o generatorima pseudoslučajnih brojeva te ćemo navesti tri primjera algoritama za takve generatore. Osim toga, reći ćemo nešto i o generatorima „pravih" brojeva, koji bolje opisuju „pravu" slučajnost. Na kraju ćemo usporediti ta dva tipa generatora te detaljnije opisati njihovu primjenu.

2 Generiranje niza slučajnih brojeva

Definicija 1 (Kolmogorov–Chaiten)**.** Niz slučajnih brojeva je *slučajan* kada je najkraći program koji ga može generirati jednako dug kao i sam niz. ◁

Intuitivno, niz brojeva je jednostavan ako ga možemo jednostavno opisati u nekoliko riječi, npr. niz brojeva 0101010101010101010101010101 sadrži 16 ponavljanja 01. Stoga, niz brojeva je složen ukoliko ne postoji takav jednostavan opis. Kada kažemo da za neki niz postoji program koji ga može generirati, pretpostavljamo da postoji *Turing*ov stroj koji može generirati taj niz.

Definicija 2. Kažemo da je program p opis niza x na *Turing*ovom stroju T ukoliko je $T(p) = x$. Duljina najkraćeg opisa definirana je sa

$$K_T(x) \coloneqq \min_p \{l(p) \colon T(p) = x\}, \tag{1}$$

gdje je $l(p)$ duljina programa p mjerena u bitovima. ◁

Definicija 3. Kažemo da je neki niz brojeva *slučajan* ukoliko su ispunjena dva uvjeta:

1. Vrijednosti u nizu su uniformno distribuirane na definiranom intervalu ili skupu.
2. Nemoguće je predvidjeti iduće vrijednosti na temelju prethodnih ili trenutnih.

◁

Preciznije, u definiciji 3 kažemo da je niz brojeva slučajan, ukoliko ne možemo naći nikakav uzorak pomoću prethodnih vrijednosti niza, na temelju kojeg bismo predvidjeli neke iduće vrijednosti. Problem kod ovih definicija je što iako nekad ne možemo naći program koji generira dani niz slučajnih brojeva, to ne znači da takav program ne postoji. Jednostavnije je za neki niz brojeva pokazati da nije slučajan. Više o problematici definicije niza slučajnih brojeva možete pročitati u knjizi [8].

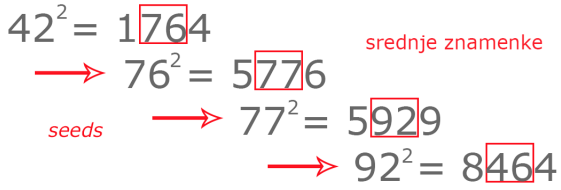
Definicija 4. *Generator slučajnih brojeva* je računalni ili fizički sustav dizajniran za generiranje niza brojeva koji se pojavljuju slučajno. ◁

Kada pričamo o generatorima slučajnih brojeva, većinom je riječ o generatorima pseudoslučajnih brojeva. Niz pseudoslučajnih brojeva nije uistinu slučajan, kako je određen s početnom vrijednošću (*seed*, izvor). Generatori pseudoslučajnih brojeva, premda nisu najbolji, važni su zbog svoje brzine generiranja velikih nizova slučajnih brojeva. Nekada se smatralo da ovi generatori daju „prave" slučajne brojeve, na što se *John von Neumann* našalio te rekao: „Svatko tko uopće pomisli koristiti aritmetičke metode za generiranje slučajnih znamenki je, naravno, u grijehu." Postoje brojni generatori pseudoslučajnih brojeva, a mi ćemo pobliže opisati nekoliko zanimljivijih.

2.1 Metoda srednjih kvadrata

Metodu je izmislio *John von Neumann* 1949. godine. Najprije opišimo algoritam:
Da bi se generirao n-znamenkasti niz pseudoslučajnih brojeva, kreira se n-znamenkasta početna vrijednost (*seed*) te se ona kvadrira, što rezultira $2n$-znamenkastim brojem. Ukoliko takav broj ima manje od $2n$ znamenki, tada se do $2n$-znamenkastog broja dolazi dodavanjem početnih nula. Srednjih n znamenki rezultata će biti idući broj u nizu i vraćen kao novi *seed*.

Ova metoda funkcionira, ako je n paran. Ukoliko je n neparan, tada „srednjih n znamenki" nije uvijek jasno definirano. Ukoliko je svaka od srednjih n znamenki jednaka 0, tada će generator beskonačno generirati nule, a ako je prva polovica broja sastavljena od znamenki 0, tada će se naredni brojevi smanjivati do nule. U praksi ova metoda nije dobra, kako će u jednom trenutku ili uvijek iznova generirati isti broj ili će se izvrtiti na neki prethodno generirani broj u nizu te zapeti u beskonačnoj petlji. Bez obzira na to, ovakva metoda je bila puno brža od čitanja „pravih" slučajnih brojeva s kartica, stoga tada, dovoljno dobra. Na slici 1 možete vidjeti primjer provođenja *von Neumann*ova algoritma.



Slika 1: Primjer provođenja algoritma

Pogledajmo također implementaciju ovog algoritma u programskom jeziku *Python*:

```
seed_number = int(input("Please enter a four digit number:\n[####] "))
number = seed_number
already_seen = set()
counter = 0

while number not in already_seen:
    counter += 1
    already_seen.add(number)
    number = int(str(number * number).zfill(8)[2:6])
    print(f"#{counter}: {number}")

print(f"We began with {seed_number}, and"
      f" have repeated ourselves after {counter} steps"
      f" with {number}.")
```

2.2 Linearni posmični generatori

Pokazat ćemo da su *xorshift* generator, kreatora *Marsaglia*e i *linear-feedback shift register* (u nastavku *LFSR*) generator, koji je osmislio *Tausworthe* veoma slični. Preciznije, pokazat ćemo, da uz određene uvjete, *xorshift* i *LFSR* generatori mogu generirati isti niz. Najprije definirajmo neke pojmove, koji će nam pomoći pri razumijevanju dokaza.

Napomena 5. Definiramo $\mathbb{F}_2$ kao konačno polje s dva elementa $\{0, 1\}$. Operacije u polju zapisujemo kao $+$ i $\times$. Ukoliko nam 0 predstavlja laž, a 1 istinu, tada su operacije u polju „isključivo ili" (`xor` ili $\oplus$) te „i" ($\wedge$).

Definicija 6. Neka je $A \in \mathbb{F}_2^{n\times n}$ matrica reda n nad poljem $\mathbb{F}_2$. *Karakteristični polinom* $k_A(z)$ matrice A definiramo sa

$$k_A(z) = \det(A - zI). \tag{2}$$

◁

Teorem 7 (Hamilton–Cayley)**.** *Neka je $A \in \mathbb{F}_2^{n\times n}$ matrica reda n nad poljem $\mathbb{F}_2$. Tada A poništava svoj karakteristični polinom, tj.*

$$k_A(A) = 0. \tag{3}$$

◎

Definicija 8. *Minimalni polinom* od A je jedinstveni normirani polinom $\mu_A(z)$ najmanjeg stupnja takav da je $\mu_A(A) = 0$ (kojeg A poništava). ◁

Napomena 9. Očito $\mu_A(z)$ dijeli $k_A(z)$.

Definicija 10. Neka je A regularna. *Period* matrice A jest najmanji pozitivni cijeli broj ρ takav da je $A^p = I$. ◁

Napomena 11. Iz teorema 7 slijedi da bilo koja pozitivna potencija matrice A može biti prikazana kao linearna kombinacija elemenata skupa $\{I, A, A^2, A^3, \ldots, A^{n-1}\}$ te da postoji najviše $2^n - 1$ netrivijalnih mogućnosti. Dakle, $\rho \leq 2^n - 1$. Maksimalni period $\rho = 2^n - 1$ se dobije ako i samo ako je minimalni polinom $\mu_A(z)$ ireducibilni polinom stupnja n.

Definicija 12. *LFSR niz* jest niz (x_j) koji zadovoljava linearnu rekurziju

$$\sum_{k=0}^{d} \alpha_k x_{j-k} = 0, \text{ za sve } j \geq d, \tag{4}$$

gdje su $\alpha_0, \alpha_1, \ldots, \alpha_d \in \mathbb{F}_2$ te pretpostavljamo da je $\alpha_0 = 1$. Takva rekurzija definira x_j kao linearnu kombinaciju elemenata $x_{j-1}, \ldots, x_{j-d}$. Ukoliko su $x_0, x_1, \ldots, x_{d-1}$ dani kao početni uvjeti, tada su svi x_j, za $j \geq d$ jedinstveno definirani rekurzijom. ◁

Linearna kombinacija x_j može biti samo jedan bit ili cijela riječ. Mi ćemo pretpostaviti da x_j mogu biti skalari ili vektori bilo kakve fiksne veličine. Tada su koeficijenti $\alpha_0, \alpha_1, \ldots, \alpha_d$ (*feedback* koeficijenti) jedinstveno reprezentirani *LFSR* polinomom povezanosti (*feedback* polinomom)

$$P(z) = \sum_{k=0}^{d} \alpha_k z^k, \tag{5}$$

koji odgovara rekurziji (4) te je pripadna funkcija izvodnica jednaka $G(z) = \sum_{m=0}^{\infty} x_m z^m$. Ukoliko funkciju izvodnicu promatramo kao formalni red, tada imamo $G(x) = \frac{P_0(z)}{P(z)}$, gdje je $P_0(z)$ polinom stupnja ne većeg od $d - 1$, ovisno o početnim uvjetima. Ukoliko je $P(z)$ ireducibilan polinom stupnja d i $P_0(z) \neq 0$, tada je niz (x_j) periodičan s periodom $2^d - 1$. Povežimo sada niz generiran *LFSR* generatorom s nizom generiranim s *xorshift* generatorom. Neka je $\beta \in \mathbb{F}_2^{1\times n}$ netrivijalni vektor čije su komponente u $\mathbb{F}_2$. Vektor β možemo promatrati kao *seed* za *xorshift* generator. Neka je $T \in \mathbb{F}_2^{n\times n}$ regularna matrica reda n nad poljem $\mathbb{F}_2$. Tada je pseudoslučajni niz n-bitnih vektora definiran sa

$$x_j = \beta T^j \tag{6}$$

te je generiran koristeći rekurziju $x_j = x_{j-1}T$, za $j \geq 1$, gdje je $x_0 = \beta$. S prikladnim odabirom matrice T, dobijemo 32-bitne i 64-bitne generatore. *Marsaglia* je imao ideju uzeti

$$T = (I + L^a)(I + R^b)(I + L^c), \tag{7}$$

gdje su

$$L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \end{bmatrix} \tag{8}$$

$$R = L^T \tag{9}$$

redom lijeve i desne posmične matrice takve da vrijedi

$$(v_1 v_2 \dots v_{n-1} v_n)L = (v_2 v_3 \dots v_n 0) \tag{10}$$
$$(v_1 v_2 \dots v_{n-1} v_n)R = (0 v_1 \dots v_{n-2} v_{n-1}) \tag{11}$$

gdje je (a, b, c) uređena trojka cijelih brojeva. T je dobro odabran ukoliko je njegov period najveći mogući, tj. $\rho = 2^n - 1$. Drugim riječima, $T^\rho = I$, a $T^j \neq I$, za $0 < j < \rho = 2^n - 1$. Iz napomene 11 možemo zaključiti da je to istina samo ukoliko je minimalni polinom od T ireducibilni polinom stupnja n. Neka je stoga $\mu_T(z) = \sum_{k=0}^{d} \alpha_k z^{d-k}$. Možemo pretpostaviti da je $\mu_T(z)$ stupnja $d \leq n$ te da je $\alpha_0 = 1$ pa je

$$\sum_{k=0}^{d} \alpha_k T^{d-k} = 0. \tag{12}$$

Pomnožimo li slijeva sa βT^{j-d}, imamo

$$\sum_{k=0}^{d} \alpha_k \beta T^{j-k} = 0, \text{ za sve } j \geq d. \tag{13}$$

Kako znamo da je $x_j = \beta T^j$ (jer je β *seed*), slijedi da je

$$\sum_{k=0}^{d} \alpha_k x_{j-k} = 0, \text{ za sve } j \geq d. \tag{14}$$

Vidimo da je ovo zapravo linearna rekurzija (4) koju smo već promatrali. Stoga možemo zaključiti da promatrani niz (x_j) može generirati i *LFSR* čiji je *feedback* polinom jednak $\widetilde{P}(z) = \sum_{k=0}^{d} \alpha_k z^k$ [7]. Pokazali smo da *xorshift* i *LFSR* generator mogu generirati iste nizove, međutim, ovi generatori ipak se razlikuju. Implementacija *xorshift*a zahtjeva manje memorije te `xor` i `shift` operacija od standardne implementacije *LFSR* generatora, što ga čini bitno bržim.

2.3 Usporedba *PRNG* i *TRNG*

Osim generatora pseudoslučajnih brojeva (*PRNG*), postoje i generatori „pravih" slučajnih brojeva (*TRNG*). Takvi generatori izvode slučajnost iz fizičkih fenomena, poput varijacije u pokretima miša ili vrijeme između pritiska dvije tipke na tipkovnici. Atmosferski i termalni šum, elektromagnetski i kvantni fenomeni, kao i radijacija i radioaktivno raspadanje mjereno u kratkim vremenskim intervalima, mogu biti dobri izvori (*seeds*) prirodne entropije. Generatori „pravih" slučajnih brojeva su, za razliku od generatora pseudoslučajnih brojeva, nedeterministički i aperiodički, ali su i bitno sporiji. Usporedbu svojstava ova dva tipa generatora slučajnih brojeva možete vidjeti u tablici 1.

Tablica 1: Usporedba *PRNG* i *TRNG*

Svojstva	**Generatori pseudoslučajnih brojeva**	**Generatori „pravih" brojeva**
učinkovitost	izvrsno	loše
determinizam	deterministički	nedeterministički
periodičnost	periodički	aperiodički

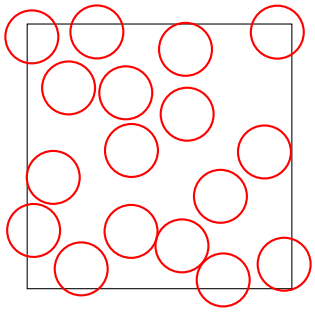
Osim po svojstvima, možemo ih usporediti i ovisno o području primjene. Kako je *TRNG* nedeterministički te je sporiji od *PRNG*, manje je prikladan za simulacije i modeliranje, gdje se radi s velikim brojem podataka, koje je bi *PRNG* puno brže izgenerirao. Međutim, u tablici 2 možete vidjeti da *TRNG* ipak nalazi svoje široko područje primjene.

Tablica 2: Područje primjene *PRNG* i *TRNG*

Primjena	**Najprikladniji generator**
lutrije	TRNG
igre i kockanje	TRNG
slučajni uzorci	TRNG
sigurnost	TRNG
simulacije	PRNG

3 Primjena generatora slučajnih brojeva

Već u samom LaTeXu možemo naići na slučajne brojeve. Postavljanjem izvora na neki broj i koristeći LaTeXov generator pseudoslučajnih brojeva, dobili smo sliku 2 koja prikazuje nepreklapajuće kružnice čija središta imaju slučajnu poziciju unutar zadanog pravokutnika.



Slika 2: Kružnice čija je pozicija slučajna

Kao što smo već spomenuli, generatori slučajnih brojeva primjenjuju se u simulacijama. Ukoliko simuliramo prirodne procese, htjeli bismo da oni budu što vjernije prikazani. To se postiže uz pomoć slučajnih brojeva koji su uvelike pridonijeli realističnijim prikazima u području nuklearne fizike i operacijskog istraživanja. U kriptografiji slučajni brojevi igraju izuzetno veliku ulogu. Važnost društvenih mreža i sigurnosti danas je veća nego ikad. Svakom korisniku raznih usluga na internetu važna je privatnost i zaštita osobnih podataka. Stoga je u kriptografiji nužno osigurati neprevidivost brojeva u generiranom nizu. Ulaz u generator mora biti odabran tako da ga je skoro pa nemoguće otkriti nekom metodom pogađanja. Uz to, period generatora mora biti velik, kako bi se postigao velik broj različitih nizova koje je nemoguće pretražiti u konačnom vremenu. Ukoliko je ulaz prikladno odabran te je period dovoljno velik, tada se takav niz može koristiti npr. kao tajni ključ za slanje poruka. Generatori slučajnih brojeva koriste se i u numeričkoj analizi. Jednostavan primjer je približno računanje površine kruga. Kockanje, kartanje i računarske igre su možda ipak najpopularniji primjeri primjene generatora slučajnih brojeva. Odatle i potiče naziv poznatog *Monte Carlo* algoritma koji također koristi slučajne brojeve.

4 Zaključak

U ovom radu upoznali ste se s osnovnim pojmovima vezano uz slučajne brojeve, nizove te generatore slučajnih brojeva. Tema je veoma aktualna te je brojnim znanstvenicima cilj osmisliti što bolji, sigurniji i brži generator slučajnih brojeva. Vidjeli smo da postoje brojni generatori pseudoslučajnih brojeva, a da isto tako postoje brojne ideje za stvaranje novih i boljih generatora „pravih" slučajnih brojeva. Generatori su važni u raznim područjima, od kojih je bitno spomenuti kriptografiju, kockarnice i računalne igre. Rastom tehnologije i interesa ljudi za zaštitu svoje privatnosti, generatori slučajnih brojeva dobivaju na važnosti, stoga će biti zanimljivo vidjeti koja nova prirodna ili fizička pojava bi mogla poslužiti kao izvor raznim generatorima.

Literatura

[1] URL: https://www.random.org/randomness/.

[2] URL: http://faculty.rhodes.edu/wetzel/random/intro.html.

[3] URL: https://en.wikipedia.org/wiki/Pseudorandom_number_generator.

[4] URL: https://en.wikipedia.org/wiki/Middle-square_method.

[5] URL: http://sigurnost.zemris.fer.hr/random/2001_juric/.

[6] URL: http://www.scholarpedia.org/article/Algorithmic_complexity.

[7] Richard Brent. „Note on Marsaglia's Xorshift Random Number Generators". *Journal of Statistical Software* 11 (srpanj 2004). DOI: 10.18637/jss.v011.i05.

[8] Gregory J. Chaitin. *Exploring Randomness*. Springer-Verlag London Ltd., 2001. ISBN: 978-1-4471-1085-9.

[9] Maja Mendler. „Generatori slučajnih brojeva i njihova primjena u kriptografskim sustavima". Mag. rad. Sveučilišta J. J. Strossmayera u Osijeku, 2015. URL: https://repozitorij.etfos.hr/islandora/object/etfos%3A491/datastream/PDF/view.

[10] Mario Skočić. „Generiranje pseudoslučajnih brojeva i testovi slučajnosti". Mag. rad. Sveučilište u Zagrebu, 2017. URL: https://repozitorij.pmf.unizg.hr/islandora/object/pmf%3A3861/datastream/PDF/view.

[11] Igor Urbiha. *Generiranje niza pseudoslučajnih brojeva*. Matematičko-fizički list, LXII2. 2010. URL: https://bib.irb.hr/datoteka/554519.MFL_2010-2011_2_Generiranje_niza_pseudoslucajnih_brojeva.pdf.