

GLAVA

11

Randomizirani algoritmi

U ovom poglavlju izučavamo randomizirane algoritme koji mogu da na slučajni način biraju sledeći korak za izvršavanje prilikom obrade ulaznih podataka. Da bi slučajno odabrali narednu granu izvršavanja, podrazumevamo da ovi algoritmi to rade na osnovu slučajnog izbora elementa iz neke kombinatorne strukture kao što su skupovi ili grafovi. Tipično, ovi algoritmi mogu izabrati slučajni element iz datog skupa, ili izabrati slučajnu granu grafa sa datim osobinama, ili generisati slučajnu permutaciju od svih mogućih permutacija date dužine. Ne samo to, sve ove operacije se smatraju osnovnim operacijama koje se izvršavaju za neko jedinično vreme.

Uopšteno govoreći, neki algoritam nazivamo *randomiziranim* ako je njegovo izvršavanje određeno ne samo ulaznim podacima, nego i slučajnim ishodima nekog eksperimenta. Ovi ishodi se obično simuliraju „slučajnim” brojnim vrednostima koje se dobijaju kao rezultati determinističkih algoritama. Pri tome, ovi brojevi se statistički ne razlikuju od pravih slučajnih brojeva, pa se zato nazivaju pseudoslučajni brojevi. Naravno, dodatno se podrazumeva da se ovakvi brojevi koji izgledaju slučajno mogu pojedinačno generisati za konstantno vreme.

Važno je razumeti da randomizirani algoritmi i probabilistička analiza algoritama nisu isti koncept. U probabilističkoj analizi se deterministički algoritam analizira pretpostavljajući slučajne ulazne podatke. Na primer, za problem sortiranja niza možemo analizirati deterministički algoritam *quick-sort* (tj. kada je izbor pivot elementa unapred određen) za ulazne podatke koji su slučajna permutacija od n elemenata. Nasuprot tome, randomizirani algo-

ritmi koriste slučajne ishode za izbor grana izvršavanja, što se svodi na izbor determinističkog algoritma iz kolekcije algoritama. Pored toga, randomizirani algoritmi obezbeđuju nezavisnost od „najgorih” ulaznih podataka.

Zašto bi bilo korisno imati algoritme koji se ponašaju slučajno? Dve glavne prednosti randomiziranih algoritama su brzina i jednostavnost. Za mnoge probleme, randomizirani algoritmi su brži od poznatih najboljih determinističkih algoritama. Pored toga, mnoge randomizirane algoritme je lakše razumeti i programski realizovati od determinističkih algoritama slične efikasnosti. (Zbog jednostavnosti su randomizirani algoritmi često i pouzdaniji od svojih determinističkih pandana.) Prema tome, za mnoge probleme su randomizirani algoritmi najprostiji ili brži ili oboje.

Pre nego što ilustrujemo neke osnovne principe dizajna i analize randomiziranih algoritama, razmotrićemo problem generisanja slučajnih ishoda koji se koriste u randomiziranim algoritmima. Naime, postavlja se pitanje kako simulirati slučajnost u okviru inherentno determinističkih računara na kojima se izvršavaju randomizirani algoritmi. Da bismo odgovorili na ovo pitanje, u narednom odeljku ćemo posvetiti veću pažnju generisanju pseudo-slučajnih brojeva. Pri tome se ograničavamo samo na uniformnu raspodelu ovih brojeva i nećemo se baviti opštijim problemom drugih raspodela verovatnoće.

11.1 Slučajni brojevi

Simuliranje stohastičkih procesa na računarima zahteva neki izvor slučajnosti. Na primer, pretpostavimo da treba da simuliramo bacanje fer kocke. To možemo postići najpre uzimanjem broja sekundi aktuelnog tačnog vremena, zatim deljenjem tog broja sa 10 i, na kraju, dodavanjem 1. Jedan od nedostataka ovog pristupa je to što ga ne možemo koristiti ukoliko eksperiment treba da ponovimo više puta. Naime, dobili bismo isti rezultat višestrukog bacanja kocke u intervalu od 10 sekundi, što je veliki vremenski interval za računare, i zato bi takav eksperiment izgledao vrlo neslučajno. Dakle, ovaj metod nije mnogo praktičan za računarske primene, naročito zbog toga što je obično potrebno generisati milione slučajnih brojeva. Ono što nam treba je dugačak niz brojeva sa istim osobinama kao niz slučajnih brojeva. U ovom odeljku ćemo opisati dva metoda za generisanje nizova brojeva koji *imitiraju* prave slučajne brojeve.

U praksi, brojevi koji izgledaju slučajni se na računaru generišu potpuno determinističkim algoritmima i zato to nikako ne mogu biti slučajni brojevi.

Drugim rečima, ako bi se ovi algoritmi izvršili u drugo vreme ili na drugim računarima sa istim početnim stanjem internih podataka, dobijeni niz brojeva bi bio potpuno isti. To je razlog zbog kojeg se ovi brojevi nazivaju **pseudoslučajni brojevi**, a algoritmi koji ih proizvode se nazivaju **generatori pseudoslučajnih brojeva**. Ipak, uz malu nepreciznost ćemo izraze „slučajni” i „pseudoslučajni” koristiti ravnopravno, osim kada ih je važno razlikovati.

Pravu slučajnost je nemoguće dobiti na determinističkom računaru, pa se zato zadovoljavamo generatorom pseudoslučajnih brojeva takvim da dobijeni niz brojeva prolazi većinu statističkih testova slučajnosti. Svi generatori pseudoslučajnih brojeva padaju na nekom statističkom testu, ali loši generatori padaju na prostim testovima i dobri generatori padaju na komplikovanim testovima. Pošto ne postoji univerzalna kolekcija testova čiji prolaz garantuje da je dati generator potpuno pouzdan, generatori pseudoslučajnih brojeva se konstruišu na osnovu ozbiljne matematičke analize njihovih strukturalnih osobina.

Pretpostavimo da nam trebaju slučajni prirodni brojevi u intervalu između 0 i nekog pozitivnog celog broja m , uniformno raspoređeni. Kod **uniformne raspodele**, svi prirodni brojevi u datom intervalu su jednakoverovatni. Da bismo proizveli slučajne realne brojeve koji su uniformno raspoređeni u intervalu $[0, 1]$, ove slučajne prirodne brojeve možemo jednostavno podeliti sa m . Većina drugih raspodela (normalna, geometrijska, Puasonova) može se dobiti na osnovu uniformne raspodele primenom odgovarajućih transformacija na uniformne slučajne brojeve. Pošto to izlazi iz okvira ove knjige, pažnju ćemo posvetiti samo generatorima **uniformnih** slučajnih brojeva.

Linearni kongruentni generator

Jedan od najprostijih i najčešće korišćenih uniformnih generatora je **linearni kongruentni generator** (ili kraće **LKG**). To svakako nije i najbolji generator, ali je zadovoljavajući u primenama u kojima se traži samo dobra aproksimacija niza slučajnih brojeva. Za početnu vrednost x_0 , ($0 < x_0 < m$), LKG izračunava niz prirodnih brojeva x_1, x_2, \dots između 0 i $m - 1$ na osnovu rekurentne formule

$$x_{i+1} = ax_i \pmod{m},$$

za $i = 0, 1, \dots$, gde su a ($1 < a < m$) i m pažljivo izabrani prirodni brojevi. U stvari, ovo se naziva **multiplikativni linearni kongruentni generator**, jer najopštiji linearni kongruentni generator sadrži dodatnu celobrojnu konstantu b i dat je izrazom $x_{i+1} = ax_i + b \pmod{m}$. Pošto se u praksi skoro uvek uzima da je $b = 0$, ovaj specijalni slučaj zvaćemo kraće LKG.

Broj x_0 od koga počinje niz se naziva *seme*. Ako je $x_0 = 0$, tada niz nije slučajan jer se dobijaju sve nule. Ako je m prost broj, tada članovi niza x_i nikad nisu 0. Na primer, ako su $m = 11$, $a = 3$ i seme $x_0 = 1$, tada se generišu brojevi

$$3, 9, 5, 4, 1, 3, 9, \dots$$

Kada se prvi put neki broj generiše po drugi put (broj 3 u ovom slučaju), dobijamo podniz koji se ponavlja. U prethodnom primeru, dužina podniza koji se stalno ponavlja je 5 brojeva; to se naziva *period* celog niza. U opštem slučaju, pošto imamo m mogućih brojeva za članove niza, najduži mogući period je m . Ako je m prost broj, tj. 0 se nikad ne dobija, maksimalni period je $m - 1$. Na primer, ako su $m = 11$, $a = 7$ i seme $x_0 = 1$, tada se generišu brojevi

$$7, 5, 2, 3, 10, 4, 6, 9, 8, 1, 7, 5, \dots$$

U ovom slučaju, dužina podniza koji se ponavlja je $m - 1 = 10$ brojeva, što je najbolja moguća vrednost.

Ako je m prost broj, razne vrednosti za a daju pun period $m - 1$ (nezavisno od vrednosti semena). Takvi linearni kongruentni generatori se nazivaju *linearni kongruentni generatori punog perioda*. Da bi se napravio LKG punog perioda koristeći 32-bitnu celobrojnu aritmetiku, m se bira tako da to bude veliki 31-bitni prost broj kako bi period bio dovoljno velik za većinu primena. Označeni 32-bitni broj leži u intervalu između -2^{31} i $2^{31} - 1$. Srećom, najveći mogući pozitivni ceo broj $2^{31} - 1 = 2\,147\,483\,647$ jeste prost broj, pa on predstavlja idealan izbor za m . Za ovaj konkretan prost broj, neke vrednosti za a kao što su $a = 16807$ ili $a = 48271$ imaju prednost nad ostalim, jer tako dobijeni linearni kongruentni generatori imaju tri osobine koje se inače traže od dobrih generatora:

1. LKG treba da bude punog perioda;
2. Generisani niz brojeva mora proći nekoliko statističkih testova slučajnosti;
3. Niz brojeva treba da se efikasno izračunava koristeći 32-bitnu aritmetiku bez prekoračenja.

Treba naglasiti da su ovi generatori vrlo osetljivi, jer male promene parametara mogu potpuno pokvariti njihovu „slučajnost”. Na primer, LKG oblika

$$x_{i+1} = 48271x_i + 1 \pmod{2^{31} - 1},$$

11.1. Slučajni brojevi**361**

izgleda možda nekako više slučajan. Međutim, pošto je

$$x_{i+1} = 48271 \cdot 179424105 + 1 \pmod{2^{31} - 1} = 179424105,$$

ako za seme uzmemo broj 179424105, dobijamo generator čiji je period 1! Prema tome, bez neke preke potrebe za promenom, treba koristiti generatore koji su dobro proučeni.

Treća od navedenih osobina na prvi pogled izgleda nevažna, jer bismo mogli reći da prekoračenje prilikom izračunavanja zapravo doprinosi slučajnosti rezultata. Ipak to nije tako, jer inače ne bismo imali garanciju punog perioda. Nažalost, za aktuelni pseudoslučajni broj s , ako bismo sledeći broj u u nizu naivno računali naredbom dodele u obliku

$$s = (a * s) \pmod{m};$$

gotovo sigurno bismo višestrukim množenjem dobili prekoračenje rezultata kod većine računara koji rade sa 32-bitnim celim brojevima. Ali se može pokazati da je izračunavanje izraza $x_{i+1} = ax_i \pmod{m}$ bez prekoračenja izvodljivo, i to malom izmenom redosleda računjanja.

Da bismo ovo pokazali, neka su q i r količnik i ostatak pri celobrojnom deljenju m sa a , tj. $q = \lfloor m/a \rfloor$ i $r = m \pmod{a} = m - aq = m - a \lfloor m/a \rfloor$. Tada,

$$\begin{aligned} x_{i+1} &= ax_i \pmod{m} \\ &= ax_i - m \lfloor ax_i/m \rfloor \\ &= ax_i - m \lfloor x_i/q \rfloor + m \lfloor x_i/q \rfloor - m \lfloor ax_i/m \rfloor \\ &= ax_i - m \lfloor x_i/q \rfloor + m(\lfloor x_i/q \rfloor - \lfloor ax_i/m \rfloor). \end{aligned}$$

Da bismo uprostiti pisanje, za fiksirane vrednosti parametara m i a , označimo $\delta(x_i) = \lfloor x_i/q \rfloor - \lfloor ax_i/m \rfloor$. Dakle,

$$\begin{aligned} x_{i+1} &= ax_i - m \lfloor x_i/q \rfloor + m\delta(x_i) \\ &= a(x_i \pmod{q} + q \lfloor x_i/q \rfloor) - m \lfloor x_i/q \rfloor + m\delta(x_i) \\ &= a(x_i \pmod{q}) + aq \lfloor x_i/q \rfloor - m \lfloor x_i/q \rfloor + m\delta(x_i) \\ &= a(x_i \pmod{q}) + (aq - m) \lfloor x_i/q \rfloor + m\delta(x_i). \end{aligned}$$

Kako je $r = m - aq$, odavde na kraju dobijamo

$$x_{i+1} = a(x_i \pmod{q}) - r \lfloor x_i/q \rfloor + m\delta(x_i). \quad (11.1)$$

Tri sabirka na desnoj strani preuređene jednačine (11.1) imaju nekoliko dobrih osobina:

1. Prvi sabirak se uvek može izračunati bez prekoračenja. Da bismo se u to uverili, zapazimo da je $0 \leq a(x_i \pmod{q}) < m$. To sledi na osnovu činjenice da je $0 \leq x_i \pmod{q} < q$, pa je $0 \leq a(x_i \pmod{q}) < aq = m - r \leq m$.
2. Ako je $r < q$, drugi sabirak se može izračunati bez prekoračenja. Naime, ako je $r < q$, ili ekvivalentno $r/q < 1$, tada je $0 \leq r \lfloor x_i/q \rfloor \leq r \cdot x_i/q = (r/q) \cdot x_i < 1 \cdot m = m$.
3. Konačno, ako je $r < q$, tada je faktor $\delta(x_i)$ u trećem sabirku ili 0 ili 1. Preciznije, neka je $d(x_i)$ razlika prva dva sabirka, tj. $d(x_i) = a(x_i \pmod{q}) - r \lfloor x_i/q \rfloor$. Tada,

$$\delta(x_i) = \begin{cases} 0, & \text{ako } d(x_i) \geq 0 \\ 1, & \text{ako } d(x_i) < 0. \end{cases}$$

Da bismo to videli, najpre zapazimo da je $-m < d(x_i) < m$, pošto je $0 \leq a(x_i \pmod{q}) < m$ i $0 \leq r \lfloor x_i/q \rfloor < m$. Dalje, pošto je $0 \leq x_{i+1} < m$, sledi da je

$$-m + m\delta(x_i) < d(x_i) + m\delta(x_i) = x_{i+1} < m,$$

ili ekvivalentno $\delta(x_i) < 2$.

Sa druge strane,

$$0 \leq x_{i+1} = d(x_i) + m\delta(x_i) < m + m\delta(x_i) = m(1 + \delta(x_i)),$$

ili ekvivalentno $1 + \delta(x_i) > 0$, tj. $\delta(x_i) > -1$. Ova i prethodna nejednakost $\delta(x_i) < 2$ daju $-1 < \delta(x_i) < 2$, a pošto je $\delta(x_i)$ ceo broj, sledi da je $\delta(x_i)$ ili 0 ili 1. Prema tome, ako je $d(x_i) \geq 0$, mora biti $\delta(x_i) = 0$, jer bi $\delta(x_i) = 1$ impliciralo $x_{i+1} = d(x_i) + m\delta(x_i) \geq m$, što je kontradikcija. Slično, ako je $d(x_i) < 0$, mora biti $\delta(x_i) = 1$, jer bi $\delta(x_i) = 0$ impliciralo $x_{i+1} = d(x_i) + m\delta(x_i) < 0$, što je opet kontradikcija.

Na primer, za konkretne vrednosti $m = 2^{31} - 1$ i $a = 16807$, imamo da je $q = 127773$ i $r = 2836$. Stoga je $r < q$ i $-m < d(x_i) < m$, pa $d(x_i)$ pripada opsegu 32-bitnih označenih celih brojeva. Pored toga, nije čak ni neophodno računati $\delta(x_i)$, pošto se na osnovu provere razlike prva dva sabirka može zaključiti da li treći sabirak iznosi 0 ili m . Naime, ako je $d(x_i) \geq 0$, tada je $x_{i+1} = d(x_i)$; a ako je $d(x_i) < 0$, tada je $x_{i+1} = d(x_i) + m$. Prema tome, izračunavanje narednog člana x_{i+1} pomoću izraza (11.1) ne proizvodi prekoračenje međurezultata.

U nastavku je prikazan algoritam LCG(s) koji primenjuje ovaj pristup za realizaciju linearnog kongruentnog generatora. U algoritmu se izračunava

11.1. Slučajni brojevi

363

naredni pseudoslučajni broj za dati prethodni pseudoslučajni broj s , pri čemu se koriste odgovarajuće globalne konstante m i a , kao i q i r takve da je $r < q$.

```

LCG(s)
  d = a(s (mod q)) - r [s/q];
  if d ≥ 0 then
    s = d;
  else
    s = d + m;
  return s;

```

Mersenov tvister

Mersenov tvister (ili kraće *MT*) je uniformni generator pseudoslučajnih brojeva koji je stekao veliku popularnost u posljednje vreme. On je brz, robusan i ima dokazan ogroman period koji je $2^{19937} - 1$. Ova vrednost perioda objašnjava njegovo ime, jer je $2^{19937} - 1$ Mersenov prost broj.¹ Dodatno, MT generiše pseudoslučajne brojeve koji prolaze vrlo stroge statističke testove. Sve ovo se dobija uz malo povećanje složenosti — MT ima nekoliko parametara dok LKG ima samo dva (a i m), kao i uz neznatno veće memorijske zahteve — MT koristi 624 memorijske reči radnog prostora dok LKG koristi samo jednu radnu memorijsku reč.

MT generiše niz binarnih vektora dužine w , koji pojedinačno predstavljaju uniformne pseudoslučajne brojeve iz intervala od 0 do $2^w - 1$ u binarnom zapisu. Ovi vektori se identifikuju sa memorijskim rečima računara dužine w , pri čemu se podrazumeva da se bit najmanje važnosti nalazi sasvim desno. Ako binarne vektore-vrste označavamo zadebljanim slovima, kao što su \mathbf{x} i \mathbf{y} , MT se zasniva na linearnoj rekurentnoj jednačini

$$\mathbf{x}_{k+n} = \mathbf{x}_{k+m} + \mathbf{x}_{k+1} \begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_r \end{pmatrix} \mathbf{A} + \mathbf{x}_k \begin{pmatrix} \mathbf{I}_{w-r} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \mathbf{A}, \quad k = 0, 1, 2, \dots, \quad (11.2)$$

gde su \mathbf{I}_r i \mathbf{I}_{w-r} jedinične matrice reda r i $w - r$, $\mathbf{0}$ je nula matrica odgovarajućeg reda i, na kraju, \mathbf{A} je 0-1 matrica reda $w \times w$. Pre nego što objasnimo sve parametre ove rekurentne relacije, uočite najpre da nam treba n početnih vektora (semena) $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{n-1}$. Tada redom za $k = 0, 1, 2, \dots$ dobijamo ostale vektore niza $\mathbf{x}_n, \mathbf{x}_{n+1}, \mathbf{x}_{n+2}, \dots$

¹Mersenovi prosti brojevi su prosti brojevi oblika $2^p - 1$, gde je p pozitivan ceo broj. Obratite ipak pažnju na to da je $2^p - 1$ prost broj samo ukoliko je p prost broj.

Rekurentna jednačina (11.2) koja definiše Mersenov tvister koristi nekoliko pažljivo izabranih konstanti: prirodan broj n koji predstavlja red rekurentne jednačine, prirodan broj m takav da $0 \leq m < n$, prirodan broj r takav da $0 \leq r < w$ i, na kraju, 0-1 matricu \mathbf{A} reda $w \times w$.

Mada rekurentna jednačina (11.2) izgleda komplikovano, možemo je napisati na sledeći način:

$$\mathbf{x}_{k+n} = \mathbf{x}_{k+m} \oplus \left((x_{w-1}^k, \dots, x_r^k, x_{r-1}^{k+1}, \dots, x_0^{k+1}) \cdot \mathbf{A} \right), \quad k = 0, 1, 2, \dots \quad (11.3)$$

Ovde simbol \oplus označava operaciju EXCLUSIVE-OR nad bitovima, a zapis

$$(x_{w-1}^k, \dots, x_r^k, x_{r-1}^{k+1}, \dots, x_0^{k+1})$$

označava vektor dobijen spajanjem viših $w - r$ bitova vektora \mathbf{x}_k i nižih r bitova vektora \mathbf{x}_{k+1} . Primitimo da se ovaj vektor može lako izračunati najpre maskiranjem odgovarajućih delova vektora \mathbf{x}_k i \mathbf{x}_{k+1} , a zatim njihovim kombinovanjem operacijom OR nad bitovima.

Matrica \mathbf{A} ima oblik koji omogućava da se ona može brzo pomnožiti sa leve strane. Taj oblik je:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ a_{w-1} & a_{w-2} & a_{w-3} & \cdots & a_0 \end{pmatrix},$$

gde je $\mathbf{a} = (a_{w-1}, a_{w-2}, \dots, a_0)$ pažljivo izabran konstantni vektor. Nije onda teško proveriti da važi

$$\mathbf{x}\mathbf{A} = \begin{cases} \mathbf{x} \gg 1, & \text{ako } x_0 = 0 \\ (\mathbf{x} \gg 1) \oplus \mathbf{a}, & \text{ako } x_0 = 1. \end{cases}$$

gde je $\mathbf{x} = (x_{w-1}, x_{w-2}, \dots, x_0)$ proizvoljni vektor i $\mathbf{x} \gg i$ označava operaciju pomeranja vektora \mathbf{x} udesno za i bitova.

Prema tome, celokupno izračunavanje rekurentne jednačine (11.3) može se obaviti primenom isključivo operacija nad bitovima, tačnije operacijama pomeranja, EXCLUSIVE-OR, AND i OR.

Da bi se dobila dobra uniformnost generisanih vektora, svaki vektor izračunat na osnovu relacije (11.3) se dodatno množi sa desne strane regularnom

11.1. Slučajni brojevi

365

0-1 matricom \mathbf{T} reda $w \times w$. Ova takozvana matrica *uniformnosti* \mathbf{T} se implicitno definiše tako da se, za dati vektor \mathbf{x} , proizvod $\mathbf{y} = \mathbf{x}\mathbf{T}$ izračunava primenom sledećih sukcesivnih transformacija:

$$\begin{aligned}\mathbf{y} &= \mathbf{x} \oplus (\mathbf{x} \gg u) \\ \mathbf{y} &= \mathbf{y} \oplus ((\mathbf{y} \ll s) \otimes \mathbf{b}) \\ \mathbf{y} &= \mathbf{y} \oplus ((\mathbf{y} \ll t) \otimes \mathbf{c}) \\ \mathbf{y} &= \mathbf{y} \oplus (\mathbf{y} \gg l)\end{aligned}$$

Ovde su parametri u, s, t i l celi brojevi između 1 i $w - 1$, \mathbf{b} i \mathbf{c} su konstantni vektori, $\mathbf{x} \ll i$ označava operaciju pomeranja vektora \mathbf{x} ulevo za i bita, a \otimes označava operaciju AND nad bitovima.

Algoritam $\text{MT}(\mathbf{x}(0), \mathbf{x}(1), \dots, \mathbf{x}(n-1))$ za Mersenov tvister je prikazan u nastavku. U njemu se koristi globalni niz od n binarnih vektora $\mathbf{x}(0), \mathbf{x}(1), \dots, \mathbf{x}(n-1)$ veličine memorijske reči. Ovi globalni vektori predstavljaju binarni zapis neoznačenih celih brojeva i početno dobijaju vrednosti nenula vektora-semena.

```

MT( $\mathbf{x}(0), \mathbf{x}(1), \dots, \mathbf{x}(n-1)$ )
  [[ Spojiti  $w - r$  viših bitove  $\mathbf{x}(0)$  i  $r$  nižih bitova  $\mathbf{x}(1)$  ]]
   $\mathbf{y} = (x_{w-1}^0, \dots, x_r^0, x_{r-1}^1, \dots, x_0^1)$ ;
  [[ Pomnožiti  $\mathbf{y}$  sdesna matricom  $\mathbf{A}$  ]]
  if  $x_0^1 = 0$  then
     $\mathbf{y} = \mathbf{y} \gg 1$ ;
  else
     $\mathbf{y} = (\mathbf{y} \gg 1) \oplus \mathbf{a}$ ;
  [[ Po bitovima rezultatu dodati po modulu 2 srednji element  $\mathbf{x}_m$  ]]
   $\mathbf{y} = \mathbf{x}_m \oplus \mathbf{y}$ ;
  [[ Pomnožiti rezultat s desne strane matricom  $\mathbf{T}$  ]]
   $\mathbf{y} = \mathbf{y} \oplus (\mathbf{y} \gg u)$ ;
   $\mathbf{y} = \mathbf{y} \oplus ((\mathbf{y} \ll s) \otimes \mathbf{b})$ ;
   $\mathbf{y} = \mathbf{y} \oplus ((\mathbf{y} \ll t) \otimes \mathbf{c})$ ;
   $\mathbf{y} = \mathbf{y} \oplus (\mathbf{y} \gg l)$ ;
  [[ Pomeriti globalne vektore cirkularno ]]
  for  $i = 1$  to  $n - 1$  do
     $\mathbf{x}(i - 1) = \mathbf{x}(i)$ ;
   $\mathbf{x}(n - 1) = \mathbf{y}$ ;
  [[ Vratiti kao rezultat naredni pseudoslučajni broj ]]
  return  $\mathbf{y}$ ;

```

Nakon izračunavanja narednog vektora-pseudoslučajnog broja u svakoj iteraciji algoritma MT, vrednosti globalnih vektora $\mathbf{x}(0), \mathbf{x}(1), \dots, \mathbf{x}(n-1)$ se cirkularno zamenjuju za sledeću iteraciju. Tako, novi vektor $\mathbf{x}(0)$ dobija vrednost starog vektora $\mathbf{x}(1)$, novi vektor $\mathbf{x}(1)$ dobija vrednost starog vektora $\mathbf{x}(2)$, i tako dalje sve do novog vektora $\mathbf{x}(n-2)$ koji dobija vrednost starog vektora $\mathbf{x}(n-1)$. Na kraju, novi vektor $\mathbf{x}(n-1)$ dobija aktuelnu vrednost izračunatog vektora.

Prema opštoj teoriji linearnih rekurentnih jednačina, niz brojeva generisanih relacijom (11.2) ima maksimalni period $2^{nw-r} - 1$ pod nekim uslovima koji utiču na izbor parametara. Pored toga, slučajnost brojeva celog perioda ne zavisi od izbora početnih vektora, ukoliko bar jedan od njih nije nula.

Da rezimiramo, MT zavisi od dva skupa parametara:

1. *Parametri perioda* određuju period generisanog niza brojeva. To su celi brojevi w (veličina memorijske reči), n (red rekurentne jednačine), m (srednji element) i r (deobna tačka memorijske reči), kao i jedan binarni vektor \mathbf{a} (matrica \mathbf{A}).
2. *Parametri uniformnosti* određuju kvalitet uniformnosti generisanog niza brojeva. To su celi brojevi u, s, t i l , kao i dva binarna vektora \mathbf{b} i \mathbf{c} .

Radi ilustracije, u tabeli 11.1 su navedene vrednosti ovih parametara iz originalne verzije Mersenovog tvistera. Te vrednosti obezbeđuju kako pun period $2^{nw-r} - 1$ tako i dobru uniformnost generisanog niza brojeva. Vrednosti za binarne vektore \mathbf{a} , \mathbf{b} i \mathbf{c} su date u heksadecimalnom zapisu.

Parametri perioda					Parametri uniformnosti						
w	n	m	r	\mathbf{a}	u	s	t	l	\mathbf{b}	\mathbf{c}	
32	624	397	31	9908B0DF	11	7	15	18	9D2C5680	EFC60000	

TABELA 11.1: Parametri za Mersenov tvister (MT19937).