Magična matematika

Travica, Ana

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ana Travica

MAGIČNA MATEMATIKA

Diplomski rad

Voditelj rada:
izv. prof. dr. sc. Matija Kazalicki

Zagreb, srpanj 2021.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Antei, Karli i bebi

Sadržaj

Uvod

Svijet magije oduvijek je intrigirao čovječanstvo. Iako se zna da nema ničeg nadnaravnog u trikovima koje izvode mađioničari, bolje rečeno iluzionisti jer tu se radi upravo o tome - iluziji, ipak uživamo na tren odlutati u svijet u kojem nemoguće postaje moguće. Kakve to veze ima s matematikom? Zasigurno bi malo tko, razmišljajući o tome što se nalazi u pozadini nekog trika pomislio da je riječ o matematici. Vjerojatno bi se kladili na posebne rekvizite i majstorije koji gledatelja "varaju" skrećući mu pozornost na neke druge stvari dok se trik odvija, a pod "matematičke trikove" bi uglavnom uvrstili ne baš toliko zabavnu "magiju" (koja, istina, sadrži vrlo malo matematike u sebi) gdje se karte beskonačno dijele u hrpe, a gledatelji se pitaju koliko će još dugo morati pristojno sjediti. Ipak, postoji čitava lepeza zabavnih trikova s kartama, kovanicama, kockama i drugim objektima koji su bazirani upravo na netrivijalnim matematičkim principima, a koji dalje otvaraju vrata novim matematičkim spoznajama i primjenama u brojnim sferama života.

Za ilustraciju, razmotrimo Gibreathov princip[1]; on nam govori da, ukoliko imamo unaprijed složeni špil parne duljine u kojemu boje karata alterniraju (naizmjence su složene crvena karta, zatim crna karta pa ponovo crvena itd. ili obrnuto), nakon miješanja špila na određeni način koji nazivamo Gilbreathovo miješanje[2] dobivamo špil iz kojeg možemo uzimati, "otkrivati", redom po dvije karte s vrha (ili dna) špila i pritom će u svakom takvom paru jedna karta biti crvene, a druga crne boje.

Mnogo različitih kartaških trikova zasnovano je baš na Gilbreathovom principu, no ono što ga čini još važnijim jest to da on ima primjenu i izvan trivijalne matematike. Primjerice, usko je povezan s Mandelbrotovim skupom[3], a otkriveno je i da ima primjenu i u Penrose-

[1] Norman Gilbreath je američki matematičar i mađioničar zaslužan za otkriće navedenog principa 1958. godine koji je stoga i nazvan po njemu.

[2] Alternirajući špil je okrenut licem prema dolje, može ga se i presijecati proizvoljan broj puta. Potom, držeći u ruci špil okrenut licem prema dolje, uzimaju se karte, jedna po jedna s vrha i stavljaju se na stol: svaka nova na prethodnu (pritom su karte posložene na stol cijelo vrijeme i dalje okrenute licem prema dolje). Kada se na taj način podijeli od prilike polovica špila (jedna "polovica" nalazi se u ruci, a druga je na stolu), karte iz jedne polovice "uguraju se" između karata iz druge (tzv. "Riffle Shuffle").

[3] Neka je c proizvoljan kompleksni broj. Ako je niz zadan iterativno formulom $z_0 = 0$, $z_{n+1} = z_n^2 + c$ ograničen, kažemo da c pripada Mandelbrotovu skupu. U suprotnom, kažemo da c ne pripada Mandelbrotovu skupu.

ovoj teoriji popločavanja ravnine te se koristi u dizajnu računalnih algoritama za postupke sortiranja.

Postoje brojni matematički principi koji se koriste za izvođenje trikova, no u ovom radu zaustavit ćemo se na magiji koja se koristi de Bruijnovim nizovima. Iako su ime dobili po Nizozemskom matematičaru Nicolaasu Govertu de Bruijnu koji je o njima pisao 1946. godine, ovi nizovi pojavili su se puno ranije. Najraniji poznati zapis nalazi se u Pingali, sanskrtskom prozodijskom tekstu u kojem se svakom mogućem trosložnom uzorku dugih i kratkih slogova daje ime (primjerice "y" za uzorak kratki-dugi-dugi ili "m" za dugi-dugi-dugi). 1894. godine A. de Riviere postavio je problem postojanja cikličkih nizova nula i jedinica duljine 2^n koji sadrže svih 2^n binarnih kombinacija duljine n. Problem je iste godine riješila Camille Flye Sainte-Marie koja je dokazala da oni uistinu postoje. 1934. M. H. Martin dokazao je postojanje takvih nizova za alfabet proizvoljne duljine te dao algoritam za njihovu konstrukciju. Konačno, kada je 1944. Kees Posthumus postavio tezu o tome da takvih različitih binarnih nizova ima $2^{2^{n-1}-n}$, a de Bruijn ju 1946. godine dokazao, ovaj problem postao je široko poznat.

U ovom diplomskom radu koji se sastoji od četiri poglavlja, prva tri prvenstveno se oslanjaju na literaturu [1]. U prvom poglavlju pomoću kartaškog trika uvode se osnovni pojmovi vezani uz de Bruijnove nizove te daju različite metode za njihovu konstrukciju. U drugom poglavlju opisane su razne primjene de Bruijnovih nizova; one nevezane uz magičnu matematiku, da bi se u trećem ponovo vratili na kartaške trikove koji uključuju generalizaciju de Bruinjnovih nizova - univerzalne cikluse. Četvrto poglavlje predstavlja praktični dio, točnije programsku realizaciju de Bruijnovih nizova koji se, kao što je već spomenuto, nalaze u pozadini trikova opisanih u radu.

Poglavlje 1

Čarolija de Bruijnovih nizova

De Bruijnovi nizovi imaju široku primjenu; koriste se, primjerice, u istočnoindijskoj glazbi, za robotski vid, u kriptografiji za stvaranje tajnih šifri te za izvođenje trikova. Upravo tako, opisujući jedan kartaški trik, bit će objašnjen pojam de Bruijnovih nizova u ovom poglavlju. U tu svrhu, koristit će se osnovna znanja iz područja teorije grafova, konačnih polja i kombinatorike.

1.1 Uvod u de Bruijnove nizove

Primjer 1.1.1. *Publika vidi izvođača trika koji nasumično odabire pet gledatelja te im daje špil karata. Svaki od njih „presijeca" špil (razdvoji ga na dva proizvoljna dijela i zamijeni im mjesta) i dodaje špil idućem odabranom sudioniku. Kada zadnji sudionik presiječe špil, izvođač ga zamoli da uzme kartu s vrha i preda špil sudioniku od kojeg ga je dobio koji potom učini isto, sve dok se špil konačno ne nađe u rukama prvog sudionika koji također uzima kartu s vrha. Sada izvođač pokušava „pogoditi" koje su karte izvučene. Na suptilan način, on pokušava doznati koji su članovi izvukli, primjerice, crvenu kartu. Kada to dozna, izvođač može točno imenovati svaku kartu pojedinog sudionika.*

Tajna „dešifriranja trika" leži u pitanju: „koji članovi publike imaju crvenu (ili crnu) boju karte?". Svaki član može na to pitanje odgovoriti na dva načina: „imam crvenu kartu / nemam crvenu kartu (imam crnu kartu)", a s obzirom na to da je odabrano pet članova, postoji $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^5 = 32$ *moguća odgovora. Dakle, za izvedbu trika nužno je bilo imati špil od upravo* 32 *karte. Ono što publika ne zna jest da su te karte unaprijed pažljivo posložene po sljedećem pravilu (konkretno, u ovom slučaju za* $k = 5$*):*

svaki niz od $k \in \mathbb{N}$ *uzastopnih karata u špilu od* $n = 2^k$ *karata je međusobno različit s obzirom na boju karata. Ovo pravilo vrijedi ciklički*[1].

[1]Neka je s $a_0a_1...a_{n-1}$ označen neki niz. Kada kažemo da neko pravilo u tom nizu vrijedi ciklički, to znači da ono vrijedi za sve nizove oblika $a_ja_{j+1}...a_{n-1}a_0a_1...a_{j-1}$, gdje je $j = 0, 1, ..., n-1$

Primjer 1.1.2. *Kako bismo bolje razumjeli trik iz primjera 1.1.1, promotrimo primjer špila od osam karata ($n = 8$). U ovom slučaju postoji 2^3 mogućih rasporeda karata, a to su:*

RRR RRB RBR RBB BRR BRB BBR BBB

(gdje oznaku 'B' koristimo za crnu boju (black), a 'R' za crvenu (red)). Tražimo niz sastavljen od oznaka R/B takav da zadovoljava uvjet iz prethodnog primjera pri čemu je $k = 3$. Lako je provjeriti da je niz RRRBBBRB jedan takav. Zamijeni li se oznaka 'R' znamenkom '1', a 'B' znamenkom '0', dobiva se niz 11100010*.*

Definicija 1.1.3. *Neka su $k \in \mathbb{N}$ i $n \in \mathbb{N}$. De Bruijnov niz reda k nad n-članim alfabetom A je ciklički niz u kojem se svaki mogući podstring duljine k pojavljuje točno jednom. Takav niz često označavamo s $B(n, k)$, a njegova duljina je n^k.*

Niz iz primjera 1.1.2 je niz reda 3, a njegova duljina je 2^3 (alfabet se sastoji od dva znaka; *R* i *B* tj. 1 i 0).

1.2 De Bruijnov graf i metode za konstrukciju de Bruijnovog niza

Nameće se pitanje: za neki dani $k \in \mathbb{N}$, postoji li de Bruijnov niz reda k i, ako postoji, koliko takvih nizova ima te kako ih možemo naći? Jedan način da odgovorimo na dano pitanje koristi teoriju grafova.

de Bruijnov graf

Promatrat ćemo usmjereni graf[2] s n simbola koji se konstruira na temelju preklapanja nizova simbola, tj. završetak jednog fragmenta istovjetan je početku drugog. Takav graf nazivat ćemo de Bruijnov graf.

Definicija 1.2.1. *k-dimenzionalni de Bruijnov graf s n simbola (gdje su $n, k \in \mathbb{N}$) je graf koji sadrži n^k vrhova označenih svim mogućim stringovima duljine k sastavljenim od danih*

[2]Graf definiramo kao uređeni par (V, E), gdje je V skup vrhova, a E skup 2-podskupova od V koje zovemo bridovi. Petlje su bridovi koji spajaju vrh sa samim sobom. Višestruki bridovi označavaju više bridova između para vrhova. Usmjereni bridovi su oni bridovi koji imaju orijentaciju tako da idu od jednog vrha prema drugome te ih reprezentiramo uređenim parovima, a ne 2–podskupovima, dok kod višestrukih bridova E postaje multiskup. Graf koji ima usmjerene bridove zvat ćemo usmjereni graf ili digraf, a graf koji ima višestruke bridove zvat ćemo multigraf.

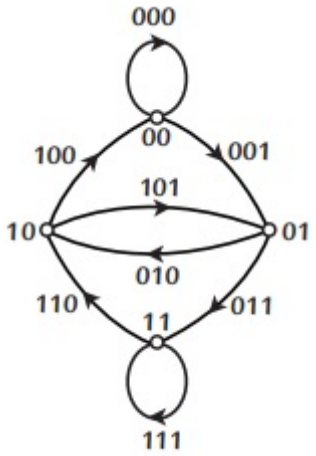
n znakova; isti simbol može se pojaviti više puta u nizu. Ako je $S := \{s_1, \ldots, s_n\}$ *skup od n znakova, onda je skup svih vrhova*

$$V = S^k = \{(s_1, s_1, \ldots, s_1), (s_1, s_1, \ldots, s_2), \ldots, (s_1, s_1, \ldots, s_n), (s_1, \ldots, s_2, s_1), \ldots, (s_n, s_n, \ldots, s_n)\}.$$

Ako jedan vrh može biti izražen pomoću drugog pomakom svih znakova za jedno mjesto u lijevo i dodavanjem novog znaka na kraj tog brida, tada potonji ima brid prema prethodnom vrhu. To znači da je skup usmjerenih bridova definiran s:

$$E = \{((t_1, t_2, \ldots, t_k), (t_2, \ldots, t_k, s_j)) : t_i \in S, 1 \leq i \leq k, 1 \leq j \leq n\}.$$

Primjer 1.2.2. *Neka je* $k = 3$*. Tada postoje* 4 *niza sastavljena od* $0/1$ *duljine* 2*, i to su:* $00, 01, 10, 11$*. Slika 1.1 pokazuje de Bruijnov graf za ove vrhove. Na primjer, postoji brid iz vrha* 01 *u* 11 *jer postoji brid* 011*, koji započinje s* 01*, a završava s* 11*. Svaki brid označen je* $0/1$ *trojkama.*



Slika 1.1: De Bruijnov graf s 4 vrha

De Bruijnov graf teško je nacrtati za velike k.

U grafu iz primjera 1.2.2 može se tražiti (usmjerena) Eulerova tura[3]. Uzmimo vrh 11 kao početni. On može sam sebe ponovo posjetiti (brid 111), zatim se posjećuju redom vrhovi: $10, 01, 10, 00, 01, 11$. Taj ciklus jednostavno se može zapisati:

[3]Kažemo za stazu da je Eulerova staza ako prolazi svim bridovima grafa. Zatvorenu Eulerovu stazu zovemo Eulerova tura.

$$11, 10, 01, 10, 00, 00, 01, 11$$

Budući da ova šetnja slijedi usmjerene bridove, svaki vrh u ciklusu ima zajednički „centar“ s idućim. Zapisujući ovaj ciklus samo tako da se dodaje nova znamenka kad se ona pojavi dobiva se de Bruijnov niz (ciklus): 11101000.

Propozicija 1.2.3. *Neka je $n = 2$. Za bilo koji $k \in \mathbb{N}$ postoji Eulerova tura na de Bruijnovom grafu koja onda definira de Bruijnov ciklus reda k.*

Dokaz. Po Eulerovom teoremu[4], povezani usmjereni graf je Eulerova tura ako i samo ako je svaki vrh parnog stupnja, tj. ima jednaki broj ulaznih i izlaznih bridova. Za de Bruijnov graf sa Slike 1.1 postoje točno dva izlazna brida za svaki vrh; $(k-1)$-torka može biti pretvorena u k-torku samo na dva načina: ili dodavanjem 0 ili dodavanjem 1. Štoviše, nije teško vidjeti da se iz svakog vrha može doći u bilo koji drugi vrh u grafu slijedeći usmjerene bridove, tj. da je graf povezan. Dakle, zadovoljeni su uvjeti Eulerovog teorema pa se može koristiti i njegov zaključak; de Bruijnov niz postoji za svaki k. Ovaj dokaz daje svojevrsni algoritam za konstrukciju: počni u bilo kojem vrhu (recimo u $k-1$ nula), odaberi bilo koji brid koji vodi iz tog vrha, obriši ga i nastavi. Dokaz pokazuje da možemo iskoristiti svaki brid točno jednom, a da ne "zaglavimo". Štoviše, dobiva se ciklus; zadnji korak završava u početnom vrhu. □

Sada se pitamo je li de Bruijnov niz jedinstven za dane $n, k \in \mathbb{N}$. Za početak, lako možemo vidjeti da Eulerova tura koju smo pronašli u Primjeru 1.2.2 nije jedina Eulerova tura u tom grafu; dobro rješenje bilo bi i primjerice 00010111. Možemo li znati koliko onda Eulerovih tura za takav graf, tj. de Bruijnovih nizova za neki fiksni $n, k \in \mathbb{N}$ ima? Odgovor na pitanje dan je u teoremu koji sljedi.

Teorem 1.2.4. *Neka su $n, k \in \mathbb{N}$. Broj različitih de Brujinovih nizova je*

$$\frac{(n!)^{n^{k-1}}}{n^k} \tag{1.1}$$

Za $n = 2$ broj različitih de Brujinovih nizova je $2^{2^{k-1}}$

Vratimo se sada na trik iz primjera 1.1.1. Jedan de Bruijnov niz reda $k = 5$ je

$$00000100101100111110001101110101.$$

[4]Eulerov teorem. (a) Multigraf bez izoliranih vrhova je Eulerov ako i samo ako je povezan, te je svaki vrh parnog stupnja. (b) Multigraf bez izoliranih vrhova ima nezatvorenu Eulerovu stazu ako i samo ako je povezan i ima točno dva vrha neparnog stupnja.

Želimo li ovaj niz iskoristiti za izvedivu verziju trika, uzet ćemo primjerice sve vrste karata od asa do „osmice“ (ukupno 32). Potom ih posložimo prateći dani niz (nebitno je koju točno vrstu karte koristimo, dokle god boje, crvena(1) i crna(0), prate niz) tako da je prva karta u nizu na vrhu, a posljednja na dnu špila:

8♣, *A*♣, 2♣, 4♣, *A*♠, 2♦, 5♣, 3♠, 6♦, 4♠, *A*♥, 3♦, 7♣, 7♠, 7♥, 6♥, 4♥, 8♥, *A*♦, 3♣, 6♣, 5♠, 3♥, 7♦, 6♠, 5♥, 2♥, 5♦, 2♠, 4♦, 8♠, 8♦

Ovako posloženi špil može biti presječen proizvoljan broj puta; ciklički uzorak de Bruijnovog niza time neće biti narušen, jedino će se početna karta promijeniti. Slika 1.2 pokazuje sve moguće kombinacije. Kada izvođač sazna tko ima crvenu kartu, koristi tablicu kako bi detektirao koje su karte točno izvučene od strane svakog pojedinog sudionika. Primjerice, ako je treći, četvrti i peti sudionik izvukao crvenu kartu, izvođač suptilno pronalazi redak tablice označen s 00111 i iščitava da su izvučene redom karte: 7♣, 7♠, 7♥, 6♥, 4♥.

00000	8♣, A♣, 2♣, 4♣, A♠	01000	8♠, 8♦, 8♣, A♣, 2♣
00001	A♣, 2♣, 4♣, A♠, 2♦	01001	A♠, 2♦, 5♣, 3♠, 6♦
00010	2♣, 4♣, A♠, 2♦, 5♣	01010	2♠, 4♦, 8♠, 8♦, 8♣
00011	3♣, 6♣, 5♠, 3♥, 7♦	01011	3♠, 6♦, 4♠, A♥, 3♦
00100	4♣, A♠, 2♦, 5♣, 3♠	01100	4♠, A♥, 3♦, 7♣, 7♠
00101	5♣, 3♠, 6♦, 4♠, A♥	01101	5♠, 3♥, 7♦, 6♠, 5♥
00110	6♣, 5♠, 3♥, 7♦, 6♠	01110	6♠, 5♥, 2♥, 5♦, 2♠
00111	7♣, 7♠, 7♥, 6♥, 4♥	01111	7♠, 7♥, 6♥, 4♥, 8♥
10000	8♦, 8♣, A♣, 2♣, 4♣	11000	8♥, A♦, 3♣, 6♣, 5♠
10001	A♦, 3♣, 6♣, 5♠, 3♥	11001	A♥, 3♦, 7♣, 7♠, 7♥
10010	2♦, 5♣, 3♠, 6♦, 4♠	11010	2♥, 5♦, 2♠, 4♦, 8♠
10011	3♦, 7♣, 7♠, 7♥, 6♥	11011	3♥, 7♦, 6♠, 5♥, 2♥
10100	4♦, 8♠, 8♦, 8♣, A♣	11100	4♥, 8♥, A♦, 3♣, 6♣
10101	5♦, 2♠, 4♦, 8♠, 8♦	11101	5♥, 2♥, 5♦, 2♠, 4♦
10110	6♦, 4♠, A♥, 3♦, 7♣	11110	6♥, 4♥, 8♥ A♦, 3♣
10111	7♦, 6♠, 5♥, 2♥, 5♦	11111	7♥, 6♥, 4♥ 8♥ A♦

Slika 1.2: Lista svih mogućih uzoraka karata za niz

Pohlepni algoritam

De Bruijnovi grafovi pokazuju da, u principu, uvijek možemo naći de Bruijnov niz te nam daju jednu konkretnu metodu za njegovu konstrukciju, međutim, pokazat će se kako postoji još mnogo različitih konstrukcija za različite primjene. Jedna od tih metoda je „pohlepni

algoritam“; počne se zapisivanjem k nula (gdje je k red de Bruijnovog niza) i dodaje se jedinicu kad god je to moguće, točnije, dokle god se ne stvori k-člani niz kakav je već sadržan u nizu.

Primjer 1.2.5. *Neka je $n = 2$ i $k = 4$. Ispišemo listu svih 4-permutacija i pratećи korake pohlepnog algoritma, uklanjamo one koje smo „iskoristili“. Na kraju dobivamo de Bruijnov niz*

$$0001111011001 00.$$

Neka je $n = 2$. Dokazano je da, ukoliko se prate koraci pohlepnog algoritma, na kraju se uvijek dobije de Bruijnov niz za bilo koji $k \in \mathbb{N}$.

Linearni pomak

Želimo izvesti trik bez „šalabahtera“ poput onog danog slikom 1.2. Prvo zapisujemo vrijednosti karata (od as-a do 8) u binarnom brojevnom sustavu; s 111 je dana „sedmica“, a „osmica“ se zapisuje kao 000. Ostale vrijednosti dane su ovako:

001 je 1,
010 je 2,
011 je 3,
100 je 4,
101 je 5,
110 je 6.

Binarne znamenke nazivaju se bitovi, a uzorci nula i jedinica s kojima se radi nazivaju se 5-bitnim riječima. Zadnja tri bita koriste se za detekciju vrijednosti karte, a prva dva za vrstu. Vrste su kodirane po sljedećem pravilu:

00 za tref (♣),
01 za pik (♠),
10 za karo (♦),
11 za herc (♥).

Primijetimo kako 0 na prvoj poziciji označava crnu, a 1 crvenu boju. Primjerice, 00001 označava A♣, a 11101 označava 5♥.

Pomoću ovakvog zapisa, čitajući 5-bitne riječi u de Bruijnovom nizu (za $k = 5$), izvođač može točno detektirati pojedine karte koje su izvučene. Da bi se bolje opisao postupak, uvodi se operacija zbrajanje modulo 2. Ona je verzija poznatog pravila zbrajanja „paran i paran daju paran, neparan i neparan daju paran, paran i neparan daju neparan, neparan i

paran daju neparan“. Ako se zamijeni „paran“ s 0, a „neparan“ s 1, dobiva se pravilo za modulo 2:

$$0 + 0 = 0, 1 + 1 = 0, 0 + 1 = 1, 1 + 0 = 1 \tag{1.2}$$

To pravilo daje jednostavan postupak za dobivanje idućeg uzorka iz neke početne 5-bitne riječi. Počnimo s jednom 5-bitnom riječi; označimo njene bitove s abcde. Idući bit izračunava se pomoću pravila

$$\text{a+c modulo 2} \tag{1.3}$$

Primjerice, ako počnemo s 01001, kao novi najdesniji bit tom nizu dodajemo 0 ($0+0=0$ po (1.2)) pa je sad niz oblika 010010. Sada čitamo da je prva karta označena sa zadnjih pet znamenki ovog niza; 10010 što označava 2♦. Postupak se nastavlja (zbrajamo modulo 2 prvi i treći bit u zadnjih pet bitova niza) dok sve karte, tj.svi bitovi nisu određeni.
Lako je provjeriti da je niz iz našeg primjera na kraju dan sa: 0100101100. Ovakav postupak često se primjenjuje u računarstvu i naziva se postupak linearnog pomaka.

Kako praktično primijeniti dano pravilo za izvođenje našeg trika? Izvođač može, na primjer, nakon što svaki sudionik uzme svoju kartu iz špila (koji je posložen po gore navedenom rasporedu), suptilno pogledati koja se karta nalazi na vrhu preostalog špila, prevesti je u oblik „abcde“ i primijeniti opisani postupak kako bi saznao koja karta je prethodila onoj koju je upravo pogledao. Na taj način, saznajemo karte obrnutim redoslijedom- od one koja je bila posljednja izvučena do one koja je prva izvučena. Također, na ovaj način izvođač može iz prethodno nepripremljenog špila izvući nasumce jednu kartu, prevesti je u kodni oblik *abcde* te koristeći opisani postupak složiti preostale karte u oblik pogodan za izvođenje trika.

Poglavlje 2

Primjene de Bruijnovih nizova

U ovom poglavlju bit će pokazano kako se de Bruijnovi nizovi i njihove varijacije koriste za robotski vid, u kriptografiji te u sastavljanju (i rastavljanju) isječaka DNA.

2.1 De Bruijnova polja

Zamislimo robota koji se kreće nekom stazom. On u svakom trenutku mora znati gdje se nalazi. Umjesto da stalno pamti pomicanja, ideja je markirati stazu po kojoj se robot kreće de Bruijnovim nizovima. Robot u tom slučaju jednostavno „pogleda dolje“ i dojavljuje niz nula i jedinica koje „vidi“. Praktično rješenje ovog problema zahtjeva jako dugačak de Bruijnov niz i pravilo za pretvorbu nula i jedinica u lokaciju na stazi na kojoj se robot nalazi. Pravi problem robotskog vida je dvodimenzionalan. Kako bismo bolje razumjeli što znači „dvodimenzionalan“ u smislu de Bruijnovog niza, promotrimo polje:

1	1	0	1
0	0	0	1
1	0	0	0
1	0	1	1

Ako je niz reda 2×2, prozor smješten u gornjem lijevom kutu pokazuje na $\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 0 & 0 \\ \hline \end{array}$. Pomicanjem 2×2 „prozora“ u bilo kojem smjeru, uključujući i ciklički te iza uglova, uvijek ćemo dobiti različiti uzorak, štoviše, pronaći ćemo svih mogućih 16 uzoraka (u svaku kućicu 2×2 prozora možemo smjestiti 0 ili 1 što znači da imamo $2 \cdot 2 \cdot 2 \cdot 2$ mogućnosti). Primjerice, podniz s dva redaka i dva stupaca smješten u sredini je: $\begin{array}{|c|c|}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline \end{array}$. Pomicanjem prozora za dva mjesta udesno očitavamo podniz dvodimenzionalnog de Bruijnovog niza:

1	0
0	1

. Još neki primjeri takvih podnizova su i:

1	1
1	1

(kojeg dobivamo pomicanjem iza uglova) te

0	1
1	0

. Dakle, polje dimenzija 4×4 je dvodimenzionalno de Bruijnovo polje.

Novija primjena ovih ideja manifestirana je u obliku digitalne olovke. Švedska tvrtka Anto izumila je posebni papir koji ima implementirano nevidljivo dvodimenzionalno de Bruijnovo polje. Digitalne olovke u svakom trenutku „znaju“ svoju poziciju na papiru i mogu se koristiti na mnoge čudesne načine. Točnije, papir je pokriven relativno nevidljivim uzorcima točkica čija je rezolucija 600 dpi-a ("dots per inch"). Olovka ima ugrađenu kameru i pomoću nje može učitati svoju točnu lokaciju jer je uzorak točkica na papiru jedinstven za svaki „prozor“ kamere. Ako se točkicama pridruži vrijednost 1, a prazninama 0, dobiva se jedna vrsta de Bruijnovog polja točkica.

Promotrimo sad polje neke proizvoljne veličine, recimo $s \times t$ te veličine prozora $u \times v$ i neka je zadano c različitih boja (točnije, neka je c duljina alfabeta) ($s, t, u, v, c \in \mathbb{N}$). Možemo se pitati: postoje li de Bruijnova polja za takve proizvoljne veličine? Ako da, postoji li neki eksplicitan način za njihovu konstrukciju? Koliko takvih de Bruijnovih polja postoji? Sva navedena pitanja, još su uvijek otvoreni problemi.

2.2 Primjena De Bruijnovih nizova u kriptografiji

Kriptografija je znanstvena disciplina koja se bavi proučavanjem metoda za slanje poruka u takvom obliku da ih samo onaj kome su namijenjene može pročitati. Iako se prvi elementi kriptografije pojavljuju već kod starih Grka u 5.stoljeću prije Krista, radi se o „industriji“ koja je danas vrlo rasprostranjena i važna. Na primjer, e-mailovi, bankovne i kreditne transakcije su kriptirane i hakeri cijelog svijeta pokušavaju razbiti te šifre. Također, glazbeni i video sadržaji često su kriptirani tako da ih se može koristiti tek onda kada se za to plati.

Postoji mnogo načina na koje poruka može biti kriptirana, a nemali broj njih koristi upravo Postupak 1.2; „zbrajanje modulo 2“. Želimo li tekst na engleskom prevesti u string nula i jedinica, iskoristit ćemo podatke sa Slike 2.1.

A	000001	I	001001	Q	010001	Y	011001	6	110110
B	000010	J	001010	R	010010	Z	011010	7	110111
C	000011	K	001011	S	010011	0	110000	8	111000
D	000100	L	001100	T	010100	1	110001	9	111001
E	000101	M	001101	U	010101	2	110010	.	101110
F	000110	N	001110	V	010110	3	110011	,	101100
G	000111	O	001111	W	010111	4	110100	*	101010
H	001000	P	010000	X	011000	5	110101	:	111010

Slika 2.1: Konverzija znakova engleske abecede u 0/1 stringove

Primjer 2.2.1. *Želi se šifrirati otvoreni tekst (poruka koju pošiljatelj želi poslati primatelju) „HELP“ (pomoć). Prvo se pomoću tablice dobije string* 001000 000101 001100 010000*, potom je izabran neki ključ (unaprijed dogovoreni uzorak po kojem se otvoreni tekst transformira) iste duljine kao otvoreni tekst, recimo* 011010101110100110110101 *i pomoću njega se šifrira tekst koristeći operaciju zbrajanje modulo 2 (1.2). Konačno, dobiven je šifrat:*

$$\begin{array}{rl} \textit{Otvoreni tekst} & \textit{001000000101001100010000} \\ \textit{Ključ} & \textit{011010101110100110110101} \\ \textit{Šifrat} & \textit{010010101011101010100101} \end{array}$$

Sada primatelj, uz poznavanje tajnog ključa, jednostavno može dešifrirati poruku. S obzirom na to da je $x + x = 0$ *modulo* 2 *za* $x = 0$ *ili* 1*, jednostavno se dobiva:*

$$\begin{array}{rll} \textit{Šifrat} & \textit{010010101011101010100101} & \\ \textit{Ključ} & \textit{011010101110100110110101} & \\ \textit{Otvoreni tekst} & \textit{001000000101001100010000} & = \textit{HELP} \end{array}$$

Međutim, problem je u tome što je nemoguće dešifrirati poruku bez ključa s obzirom na to da šifrat sam za sebe na znači ništa. Stoga se postavlja pitanje; kako pronaći dobar ključ? Za jednostavne primjene dovoljno je koristiti jedan standardni niz ili neku računalnu datoteku nizova, no u principu, ključ bi trebao biti jednokratan. Takav ključ generiran je nasumičnim procesima kao, primjerice, radioaktivnom bukom koju proizvodi udarac Geigerovog brojača[1]. I pošiljatelj i primatelj moraju imati kopiju ključa, no problem glasi: na koji će način pošiljatelj i primatelj taj ključ razmijeniti. Također, pošiljatelj mora sačuvati

[1]Geigerov brojač ili Geiger-Müllerovo brojilo je naprava ili mjerni instrument za otkrivanje ili detekciju ionizirajućega zračenja (radioaktivnosti), odnosno brojenje prolaska ionizirajućih čestica ili fotona

kopiju ključa pa, ako je uhvaćen, ključ može biti otkriven.

Kako je ovo povezano s kartaškim trikovima? Odgovor leži u de Bruijnovim nizovima. Umjesto jednokratnog ključa može se generirati dugački niz nula i jedinica koristeći Postupak 1.3 linearnog pomaka opisan u Poglavlju 1.2. Recimo da je niz počinje sa 0000000000001. Idući simbol dobiva se zbrajanjem modulo dva; prvo na taj način zbrojimo posljednji simbol i simbol koji se nalazi dva mjesta lijevo od njega, a potom rezultatu pribrojimo simbol koji se nalazi sedam, tj. pet mjesta lijevo od posljednjeg, tj. trećeg simbola gledajući od kraja niza. Rezultirajući niz se nastavlja 11010010... bez ciklusa idućih 2^{20} znamenaka. Pošiljatelj i primatelj se mogu dogovoriti za bilo koji početni niz; to može biti primjerice, neki značajan datum ili broj katova u nekoj zgradi. U suštini, ista shema koristi se milijune puta dnevno u mobilnim telefonima kao dio CDMA tehnologije.

Ova ideja ima puno varijacija, pogledajmo primjer jedne od njih.

Primjer 2.2.2. *De Bruijnov niz reda* 3 *oblika* 00011101 *formiran je iz* 001 *dodavanjem sume prvog i zadnjeg simbola i potom brisanjem prvog simbola sve dok se ne dođe ponovo do* 001. *Dakle;*

$$001 \to 011 \to 111 \to 110 \to 101 \to 010 \to 100 \to 001.$$

Ovdje smo dodali osam znamenaka 0 *na početku. Kako bismo ovo iskoristili kao kod, postupit ćemo onako kako je opisano u tablici 2.1.*

Otvoreni tekst	Ključ	Šifrat
000	000	000
001	011	010
011	111	100
111	110	001
110	101	011
101	010	111
010	100	110
100	001	101

Tablica 2.1: Kriptirana 3-bitna poruka

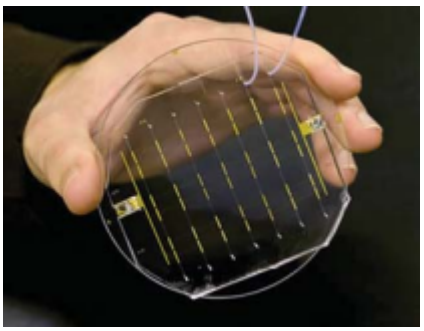
Stupac „Otvoreni tekst“ pokazuje svih osam mogućih 0/1 trojki u redoslijedu stvorenom de Bruijnovim nizom (dodali smo 000 na vrh svakog stupca). Stupac „Ključ“ prikazuje sljedeći string, također u redoslijedu stvorenim de Bruijnovim nizom. Stupac „Šifrat"

pokazuje zbroj modulo dva poruke i ključa po recima. Primijetimo da se svaki niz $0/1$ također pojavljuje samo jednom u stupcu „Šifrat".

Ovakvi su kodovi (obično nešto duži, npr duljine 5) često upisani u računalne čipove u jedinicama koje nazivamo s-boxovi. Nekoliko njih s drugačijim de Bruijnovim ciklusima često su spojeni tako da, nakon što je poruka predana jednom s-boxu, on ju šalje drugom (kojem točno, to ovisi o poruci) koji je potom šalje trećem i tako dalje, sve dok se ne generira konačni kodirani niz. Uz poman odabir de Bruijnovih nizova i pravila od box-a do box-a, ove sheme čine praktični algoritam - Data Encryption standard kojeg su izumili inženjeri IBM-a i National Security agencija. Taj standard bio je godinama korišten milijune puta dnevno za sve vrste praktičnih aktivnosti.

2.3 DNA i de Bruijnovi nizovi

DNA nije jedinstvena molekula. Radi se o paru molekula međusobno povezanih vodikovim vezama i organiziranih tako da su njihovi lanci međusobno komplementarni i antiparalelni od početka do kraja. Svaki se lanac DNA sastoji od građevnih jedinica zvanih nukleotidi kojih u DNA ima 4 vrste: adenin (A), citozin (C), gvanin (G) i timin (T). Te osnovne komponente nukleinskih kiselina mogu biti polimerizirane bilo kojim redom, npr $AACTCCAGTATGGC\ldots$. Ovi obrasci vrlo su važni i koriste se za prepoznavanje kriminalaca, utvrđivanje porijekla, razumijevanje bolesti i stvaranje lijekova. S druge strane, vrlo ih je teško pročitati i to je trenutak u kojem matematika ulazi u igru. Još uvijek ne postoji tehnologija kojom se može očitati cijeli genom od početka do kraja, već se u stanicama uzorka nalaze milijunu kopija identične DNA. Kad se one razlome i nastanu očitanja, ne zna se iz kojeg dijela genoma dolaze ta očitanja te se moraju koristiti njihovi preklapajući dijelovi da bi se rekonstruirala DNA.



Slika 2.2: Čip za sekvenciranje DNA

Slika 2.2 pokazuje čip za sekvenciranje. Radi se o 8×8 nizu s posebnim mjestom za svaki od $4 \cdot 4 \cdot 4 = 64$ uzoraka; od *AAA* do TTT. Lanac DNA koji će se sekvencionirati komunicira s poljem i svaka uzastopna trojka prisutna u niti je istaknuta u polju.

Primjer 2.3.1. *Promotrimo Sliku 2.3. Označene trojke predstavljaju dijelove nekog genoma. Problem glasi: kako iz danih naglasaka iščitati o kojem je genomu riječ? Kao što je spomenuto u uvodnom dijelu potpoglavlja; iskoristit ćemo preklapajuće dijelove.*

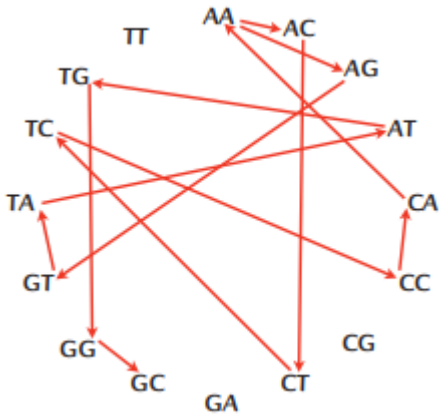
AAA	ACA	AGA	ATA	AAC	ACC	AGC	ATC
CAG	ACG	AGG	ATG	AAT	ACT	AGT	ATT
CAA	CCA	CGA	CTA	CAC	CCC	CGC	CTC
AAG	CCG	CGG	CTG	CAT	CCT	CGT	CTT
GAA	GCA	GGA	GTA	GAC	GCC	GGC	GTC
GAG	GCG	GGG	GTG	GAT	GCT	GGT	GTT
TAA	TCA	TGA	TTA	TAC	TCC	TGC	TTC
TAG	TCG	TGG	TTG	TAT	TCT	TGT	TTT

Slika 2.3: Polje trojki iz AACTCCAGTATGGC

Ispišimo najprije sve označene trojke (s obzirom na to da ne znamo njihov originalni poredak, bit će poredane po abecedi):

AAC ATG ACT AGT CAA CCA CTC CAG GCA GTA GGC TCC TGG TAT

Vrhovi grafa označavat će preklapajuće dijelove ovih trojki te će međusobno biti spojeni usmjerenim bridovima tako da prvi (izlazni) vrh predstavlja početak, a drugi (ulazni) kraj neke od danih trojki. Primjerice, s obzirom na to da je trojka AAC označena, graf za ovaj niz ima usmjereni brid iz vrha AA u vrh AC. Konačan izgled grafa dan je Slikom 2.4.



Slika 2.4: de Bruijnov graf za niz AACATTACAATCACCGA

Da bismo rekonstruirali niz iz grafa, trebamo malu varijaciju Eulerovog teorema; prethodna verzija iz Poglavlja 1 kao nužan i dovoljan uvjet za ciklus imala je stazu koja prolazi svakim bridom samo jednom te počinje i završava u istom vrhu. Za problem sekvencioniranja početni i završni vrh ne moraju biti jednaki.

Teorem 2.3.2. *Graf kojim je opisan niz počinje u vrhu x i završava u y ako i samo ako je ulazni stupanj pojedinog vrha*[2] *jednak izlaznom stupnju tog istog vrha*[3] *te to vrijedi za sve vrhove osim za početni i završni gdje je $out(x) - in(x) = 1$ i $out(y) - in(y) = -1$ (kao i prije, uvijek je moguće doći iz jednog vrha u bilo koji drugi nekim putem).*

Primjer 2.3.3. *Promatramo li graf sa Slike 2.4 vidimo da vrh AA ima ulazni stupanj* 1*, a izlazni* 2*. Dake, $out(AA) - in(AA) = 1$. Nadalje, $out(GC) - in(GC) = -1$. Ostali vrhovi imaju jednake stupnjeve ulaznih i izlaznih bridova. Po Eulerovom teoremu, postoji niz koji počinje s AA, završava s CC i sadrži sve potrebne tranzicije. Sada se može pronaći taj niz tako da se krene iz vrha AA i odabere bilo koji mogući izlazni brid te se obriše svaki brid jednom kada je iskorišten. Teorem kaže da na ovaj način nikada nećemo nigdje „zapeti“ te ćemo, kad završimo, imati stazu koja počinje u AA i završava u GC sa svim pripadajućim tranzicijskim bridovima. Tako prateći Eulerovu šetnju u danom de Bruijnovom grafu, konačno se može rekonstruirati genom dan tablicom na Slici 2.3; radi se o nizu $AACTCCAGTATGGC$.*

Ova ideja naziva se Eulerovim algoritmom i također se koristi i za mnogo kompliciranije podatke. Primjer 2.3.3 predstavlja pojednostavljeni prikaz. Za dulje nizove potrebno

[2]Ulazni stupanj vrha (engl. in-degree) je broj bridova koji završavaju u njemu. Označavamo ga s $in(v_i)$, gdje je v_i neki vrh grafa

[3]Izlazni stupanj vrha (engl. out-degree) je broj bridova koji počinju u njemu. Označavamo ga s $out(v_i)$

je napraviti nekoliko različitih rekonstrukcija koje odgovaraju dostupnim podacima. Također, kako bismo jasnije objasnili ovaj postupak, zanemareni su problemi pogrešaka u označavanju i neki drugi problemi.

Udaljenost DNA

Poznata je činjenica da ljudi i miševi dijele 85% ljudske DNA. Upravo zbog toga miševi se godinama koriste u laboratorijima kao pokusne životinje za istraživanje procesa ljudskih bolesti. Ova činjenica vodi nas do pojma udaljenosti između dvije DNA vrpce.

Neka su x i y neke dvije vrpce DNA. Funkcija udaljenosti $d(x, y)$ koristi se za mjerenje sličnosti između x i y. Ona se temelji na minimalnom broju „koraka“ koje treba napraviti da bismo x „pretvorili“ u y. Dozvoljeni koraci su: umetanje i brisanje znaka te preokretanje dijelova niza.

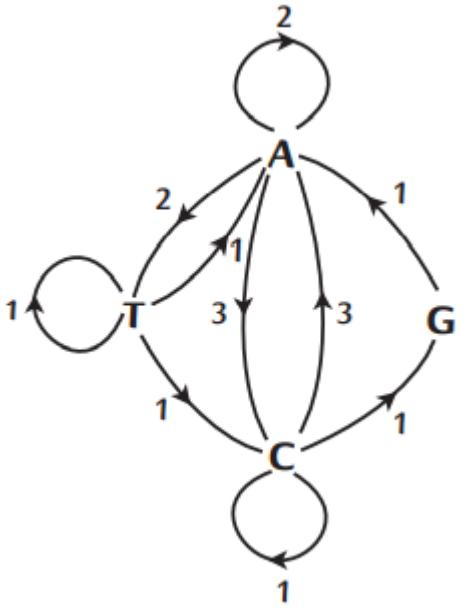
Primjer 2.3.4. $d(AAT, AAGT) = 1$ *jer je u x potrebno samo umetnuti G da bi dobili y.*

Pretpostavimo da su dane neke DNA x i y te je izračunato $d(x, y) = 137$. *Postavlja se pitanje je li to „velika“ ili „mala“ udaljenost? Da bi se to ustanovilo, ideja je konstruirati „puno“ slučajnih nizova* x' *i* y' *koji imaju istu statistiku pojavljivanja parova kao x i y, a čija će konstrukcija biti opisana u nastavku. Jednom kada su* x' *i* y' *konstruirani, može se izračunati* $d(x', y')$ *za puno takvih parova i iz toga zaključiti je li dana udaljenost (konkretno, za ovaj primjer* 137*) velika ili mala.*

Promotrimo u tu svrhu niz $x = AACATTACAATCACCGA$. *Konstruirajmo polje tranzicije za x bilježeći za svaki mogući par slova koliko se puta pojavljuje u nizu:*

	A	*C*	*T*	*G*
A	*2*	*3*	*2*	*0*
C	*3*	*1*	*0*	*1*
G	*1*	*0*	*0*	*0*
T	*1*	*1*	*1*	*0*

Postoji mnogo različitih stringova s istim poljem tranzicije. Primjerice, string AATTACACCGAATCACA ima isto polje tranzicije kao i x kojeg promatramo. Postoji li način da se spomenuti stringovi s istim poljem tranzicije, koji će poslužiti za kalibriranje originalne udaljenosti, pronađu? Odgovor je da, a ideja je jednostavna; za dano polje formira se de Bruijnov graf s četiri vrha: A, C, T i G te s usmjerenim težinskim bridovima iz jednog vrha u drugi pri čemu je težina brida ona koja je zapisana u polju. Za naš konkretan primjer de Bruijnov graf dan je Slikom 2.5.



Slika 2.5: Označeni de Bruijnov graf za AACATTACAATCACCGA

Kako bismo generirali nasumičan string koji počinje, primjerice, slovom A, jednostavno odaberemo jedan brid koji vodi iz vrha A te ga pratimo umanjujući težinu brida kojim prolazimo za 1 *i bilježimo sve vrhove kojima prolazimo tokom šetnje po grafu te tako dobivamo jedno rješenje.*

Važno je napomenuti da ne započinju svi nizovi koji imaju isto polje tranzicije istim slovom.

Pomoću ovog primjera možemo jasno vidjeti poveznicu između kartaških trikova i DNA – povezuje ih „potreba“ za de Bruijnovim grafovima.

Poglavlje 3

Univerzalni ciklusi

Ovo poglavlje vraćam se na problematiku trikova koji vode do nekih matematičkih problema koji će još dugo vremena biti neriješeni izazovi.

3.1 Važnost poretka

Primjer 3.1.1. *Promatramo trik koji počinje kao u Primjeru 1.1.1. Kada svaki sudionik izvuče svoju kartu, izvođač zamoli da se javi onaj koji je izvukao najveću, zatim drugu najveću i na kraju treću najveću te potom točno pogađa svaku od te tri karte te na kraju još preostale dvije.*

Na izvođačevu zamolbu da se jave osobe s najvećom, drugom i trećem najvećom kartom, članovi publike mogu odgovoriti na $5\times4\times3 = 60$ *načina (najveću može imati bilo tko od petero članova, zatim drugu najveću bilo tko od preostalih četvero, a treću bilo tko od preostalih troje). To je i više nego dovoljno da se odluči na kojem je mjestu od* 52 *pozicije špil "prerezan". Naravno, raspored u špilu je unaprijed poznat izvođaču i on osigurava da svaka uzastopna grupa od pet karata ima jedinstvenu oznaku „najveće, sljedeće najveće i treće najveće karte“.*

Neka je sada špil od 32 karte sastavljen od 16 crvenih i 16 crnih karata i nasumično pomiješan. Koja je vjerojatnost da je takav nasumičan raspored karata rezultirao de Bruijnovim nizom? Broj mogućih rasporeda 32 karte je

$$32! = 32 \cdot 31 \cdot \ldots \cdot 2 \cdot 1 = 263130836933693530167218012160000000$$

Iz prijašnjih poglavlja znamo da postoji $32 \cdot 2^{2^{5-1}-5} = 2^{2^4} = 2^{16}$ različitih de Bruijnovih nizova. Svaki od njih možemo označiti na $16! \cdot 16!$ načina. Konačno, vjerojatnost da u nasumično pomiješanom špilu od 32 karte pronađemo de Bruijnov niz je $\frac{16!\cdot16!\cdot2^{16}}{32!} \approx \frac{1}{9200}$. Dakle, od prilike nakon svakih 9200 miješanja očekujemo pronaći de Bruijnov niz,

a u milijun miješanja trebali bismo pronaći više od sto različitih de Bruijnovih nizova. Međutim, ovakva strategija „nasumičnog pokušaja“ pokazuje se sve manje efikasnom kako niz raste. Recimo da imamo red k i špil veličine 2^k. Šansa da će nasumično promiješan špil biti poredan kao de Bruijnov niz je

$$\frac{2^{2^{k-1}} \cdot (2^{k-1}!)^2}{2^k!}.$$

Iskoristimo li Stirlingovu aproksimaciju[1] za $n!$ vidjet ćemo da taj broj može biti dobro aproksimiran sa

$$\frac{\sqrt{\pi 2^{k-1}}}{2^{2^{k-1}}},$$

a to teži u nulu eksponencijalno brzo.

Vratimo se na primjer: k različitih objekata može biti složeno na $k! = k \cdot (k-1) \cdot (k-2) \cdot ... \cdot 2 \cdot 1$ načina. Posebno, tri različita objekta mogu biti složena na $3! = 3 \cdot 2 \cdot 1 = 6$ načina i to su:

$$123\ 132\ 213\ 231\ 312\ 321.$$

Postoji li neka permutacija znamenki $1, 2, 3, 4, 5, 6$ takva da, ako promatramo različite grupe od tri objekta, poredak svake grupe (visoka, srednja, niska) je različit (i to vrijedi i ciklički)? Jedno od rješenja je

$$1\ 4\ 6\ 2\ 5\ 3.$$

U prvoj grupi, 146, poredak je niska, srednja, visoka(NSV), iduća, 462 je SVN. Zatim VNS pa NVS, zatim ciklički, VSN i na kraju SNV.

Promotrimo sada isti problem za neki veći skup; primjerice, postoji li permutacija znamenki $1, 2, 3, ..., 24$ takva da svaka različita grupa od četiri objekta daje različiti poredak? Za sad je dokazano da uvijek postoji barem jedan način da se takav poredak nađe za proizvoljan $k \in \mathbb{N}$. To rješava prvi na listi problema, ali ostavlja otvoreno pitanje pronalaženja upotrebljive konstrukcije, približnog ili točnog broja rješenja te problem pronalaženja inverzne konstrukcije.

[1] Stirlingova formula koristi se da bi se približno izračunale faktorijele jako velikih brojeva i dana je s: $n! \approx \sqrt{2\pi n} \cdot (\frac{n}{e})^n = S\,t(n)$

3.2 Ponovljene vrijednosti i princip produkta

Ponovljene vrijednosti

Definicija 3.2.1. *Neka je dan neki špil karata. Znamo da svaka karta ima svoju vrijednost (standardno, to su: A, 1, 2, ..., K, Q, J). Ukoliko se u špilu dva ili više puta pojavljuje neka karta označena istom vrijednošću, tu vrijednost nazivamo ponovljenom vrijednošću.*

Na primjer, u regularnom špilu, sve vrijednosti su ponovljene jer on sadrži četiri as-a, četiri dvojke itd.

Primjer 3.2.2. *Neka je dan špil od* 24 *karte; pet ih je označeno s* 1*, pet s* 2*, pet s* 3*, pet s* 4 *i samo četiri s* 5*. Ako ih poslažemo u redoslijedu:* 123412534153214532413254*, onda svaka uzastopna grupa od četiri karte ima međusobno različit raspored. Može se pokazati da u špilu s po šest jedinica, dvojki, trojki i četvorki ne postoji takav raspored.*

Teorem 3.2.3. *Neka je $k \in \mathbb{N}$ red nekog de Bruijnovog niza, a njegova duljina neka je $k!$. Najmanji broj međusobno različitih vrijednosti (znamenaka) koje se pojavljuju u takvom nizu (tj.duljina alfabeta), a da je pri tom svaki podniz od k uzastopnih znamenaka u danom nizu međusobno različit, je $k + 1$.*

Napomena 3.2.4. *Dokaz ovog teorema dao je J. Robert Johnson [2].*

Permutacije s vezama

Primjer 3.2.5. *Promotrimo trik iz Primjera 3.1.1. Može se dogoditi da, kada je sudionik s najvišom vrijednošću karte zamoljen da ustane, to učini više njih. Taj slučaj otvara temu permutacija s vezama. Promatrajmo tri simbola:* $1, 2$ *i* 3*. Znamo da postoji samo šest razmještaja različitih vrijednosti, no ako su veze dopuštene, onda ih je trinaest:*

$$111, 112, 121, 211, 122, 212, 221, 123, 132, 213, 231, 312, 321.$$

Napomenimo kako je ovdje 111 *isto što i* 222 *ili* 333 *s obzirom na to da su sve vrijednosti vezane. Možemo se pitati može li se rasporediti trinaest karata tako da se u zasebnim grupama od tri karte pojave sve moguće permutacije s dopuštenim vezama. Odgovor je da, a jedan takav raspored izgleda ovako:*

$$5\ 5\ 5\ 4\ 5\ 1\ 3\ 2\ 2\ 3\ 2\ 1\ 4.$$

Koliko permutacija s vezama od k objekata postoji? Označimo taj broj s $G(k)$. $G(2) = 3$, $G(3) = 13$, $G(4) = 75$, $G(5) = 541$. Broj raste munjevito. Možemo li posložiti špil od 52 karte tako da svaka zasebna grupa od četiri predstavlja različitu permutaciju s dopuštenim vezama? Ne znamo reći sa sigurnošću, ali slutimo da je odgovor da.

Princip produkta

Teorem 3.2.6. *(Princip produkta) Neka konačni skupovi S_i imaju n_i elemenata, $n_i \in \mathbb{N}$, $i = 1, 2, ..., k$, tj. $|S_i| = n_i$. Ako je $S = S_1 \times S_2 \times ... \times S_k$ (kartezijev produkt skupova[2]), onda skup S ima $n = n_1 \cdot n_2 \cdot ... \cdot n_k$ elemenata (uređenih k-torki $s = (s_1, s_2, ..., s_k)$):*

$$|S_1 \times S_2 \times .... \times S_k| = |S_1| \cdot |S_2| \cdot ... \cdot |S_k|$$

Primjer 3.2.7. *Špil je proslijeđen i presječen mnogo puta te troje sudionika uzima različite karte. Izvođač zamoli da se sudionici s izvučenim kartama poredaju silazno po veličini karata te da istupe oni koji imaju npr. crvenu kartu. S tim informacijama izvođač može točno imenovati pojedine izvučene karte. Kako funkcionira trik? Postoji 6 mogućih rasporeda po kojima se tri sudionika mogu složiti te 8 načina za slaganje karata po crno/crvenom uzorku. Dakle, u teoriji postoji $6 \cdot 8 = 48$ odgovora. Pretpostavimo da su iz špila uklonjena četiri as-a. Možemo li posložiti preostalih 48 karata tako da svaka uzastopna grupa od tri karte daje različit odgovor? Pomoću teorije produkata možemo dokazati da produkt de Bruijnovih nizova bilo koje vrste (npr. nula/jedan, crvena/crna/plava) može biti pronađen. Rješenje problema trika sa 48 karata dano je u Tablicom 3.1.*

AH	000	123	8K	000	231	2K	000	132
6H	001	231	JK	001	312	QK	001	321
QH	010	312	3H	010	123	8H	010	213
3T	101	123	9P	101	231	3P	101	132
7K	011	231	TK	011	312	JH	011	321
8P	111	312	AP	111	132	4P	111	213
4T	110	123	7T	110	213	2P	110	132
6P	100	231	5P	100	132	6T	100	321
9K	000	312	TK	000	321	5K	000	213
2H	001	123	9H	001	213	4H	001	132
5H	010	231	AK	010	132	7H	010	321
JP	101	312	TT	101	321	5T	101	213
4K	011	123	6K	011	213	3K	011	132
TP	111	231	AT	111	123	QT	111	321
QP	110	312	8T	110	231	9T	110	321
2T	100	123	JT	100	213	7P	100	312

Tablica 3.1: Niz za „presjeci više puta, poredaj se i neka svi crveni istupe“

[2] s $S_1 \times S_2 \times ... \times S_k$ označavamo Kartezijev produkt skupova $S_1, ..., S_k : S = S_1 \times S_2 \times ... \times S_k = \{(s_1, ..., s_k) : s_i \in S_i\}$

(Napomena): prvi stupac tablice pokazuje raspored špila, drugi i treći stupac pokazuju boju i obrazac rasporeda ako je odgovarajuća karta presječena s vrha.

Primjer 3.2.8. *Špil od* 52 *karte je podijeljen na već poznati način; uključena su tri sudionika i svaki od njih presijeca špil te nakon što to svi učine, svaki izvlači kartu s vrha. Izvođač zamoli prvog sudionika da mu kaže vrijednost svoje karte, ali ne i vrstu, drugog zamoli da mu kaže vrstu, ali ne i vrijednost, a treći mu ne mora reći ništa, samo se mora „skoncentrirati" na svoju kartu. Izvođač sada raspolaže svim podacima da točno pogodi pojedinu kartu.*

Koja je pozadina ovog trika? Prvi sudionik može dati neki od 13 *odgovora, a drugi može dati* 4*; to je* $13 \cdot 4 = 52$ *mogućnosti. Stoga znamo da postoji dovoljno informacija koje nam otkrivaju gdje je špil (od* 52 *karte) presječen.*

*Postoje mnemotehnike pomoću kojih je moguće zapamtiti redoslijed karti u unaprijed posloženom špilu. Jedna od njih po kojoj se slažu vrijednosti u špilu je „Eight kings": „Eight(*8*) kings(K) threatened(*3, 10*) to(*2*) save(*7*) ninety(*9*)- five(*5*) queens(Q) for(*4*) one(*1*) sick(*6*) knave(J)." (Osam kraljeva prijetili su da će spasiti devedeset i pet kraljica za jednog bolesnog dečka). Vrste se pak mogu posložiti koristeći CHaSeD anagram (clubs(tref), hearts (herc), spades(pik), diamonds(karo)) te se one ciklički ponavljaju. Konačno se 8 tref nalazi na vrhu, a dečko karo na dnu špila. Dakle, kad nam prvi gledatelj kaže da recimo ima kartu s vrijednošću 8, a drugi kaže da ima herc-a, znamo da prvi ima osmicu od trefa (jer je tref prije herca), a drugi kralja od herca (jer kralj ide nakon 8) i konačno znamo da je treći izvukao trojku od pika (jer je nakon kralja 3, a nakon herca je pik). Bilo koji ciklički uzorak može biti iskorišten za izvođenje ovog trika.*

3.3 Univerzalni ciklusi

Sada konačno možemo dati objašnjenje naziva ovog poglavlja. Promatramo bilo koju listu prirodnih kombinatornih objekata, primjerice, stringovi nula i jedinica ili permutacije. Ideja je pronaći dugački (ciklički) niz i grupe duljine $k \in \mathbb{N}$ takav da svaka zasebna uzastopna grupa duljine k kodira jedinstveni objekt u našoj listi.

Promotrimo trik s 52 karte i tri sudionika gdje svaki sudionik imenuje vrstu karte koju je izvukao. Znamo da postoji $4 \cdot 4 \cdot 4 = 64$ moguća odgovora što je dovoljan broj mogućnosti da izvođač točno imenuje svaku izvučenu kartu; to zahtjeva niz K, P, H, T duljine 52 takav da svake tri uzastopne karte imaju različite uzorke. Ovaj trik možemo izvesti koristeći univerzalne cikluse.

Definicija 3.3.1. *Neka je dana neka familija kombinatornih objekata duljine n. Označimo je s* $\mathscr{F}_n$ *. Definirajmo* $m := |\mathscr{F}_n|$*. Pretpostavimo da je svaki* $F \in \mathscr{F}_n$ *kodiran nekim nizom* $\langle x_1, ..., x_n \rangle$*, gdje je* $x_i \in A$ *za neki fiksni alfabet A. Reći ćemo da je* $U = (a_0, a_1, ..., a_{m-1})$

univerzalni ciklus za $\mathscr{F}_n$ *(ili skraćeno U-ciklus) ako* $\langle a_{i+1}, ..., a_{i+n}\rangle$, $0 \leq i < m$ *prolazi kroz svaki element iz* $\mathscr{F}_n$ *točno jednom, pri čemu su indeksi dodijeljeni operacijom modulo n.*

Definicija 3.3.2. *Broj particija konačnog skupa od* $n \in \mathbb{N}$ *elemenata nazivamo Bellov broj i označujemo ga s* $B(n)$.

Primjer 3.3.3. *Neka je* $S = \{A, B, C\}$. *S možemo podijeliti u particije:* $\{\{A\}, \{B\}, \{C\}\}$, $\{\{A, B\}, \{C\}\}, \{\{A, C\}, \{B\}\}, \{\{A\}, \{B, C\}\}, \{\{A, B, C\}\}$. *Dakle* $B(3) = 5$.

Zanimljivo, $B(5) = 52$ što je upravo onoliko karata koliko sadrži standardni kartaški špil pa se nameće pitanje: gdje je trik?

Primjer 3.3.4. *Započnimo s* 52 *karte i pet sudionika kao i do sad: špil je proizvoljno puta presječen i pet sudionika izvlači karte s vrha špila. Izvođač ih potom zamoli da se grupiraju po vrstama karata. Sada izvođač ima dovoljno informacija da imenuje svaku pojedinu kartu. Karte bi trebale biti raspoređene tako da svakih pet uzastopnih karata imaju različite uzorke.*

Neka su izvučene sljedeće karte

član 1	*član 2*	*član 3*	*član 4*	*član 5*
8 Tref	*4 Karo*	*J Karo*	*A Herc*	*10 Tref*

Sudionici će se grupirati u particiju $\{1, 5\}\{2, 3\}\{4\}$. *Potreban nam je raspored simbola T, H, P, K u nizu duljine* 52 *takav da se svaka particija pojavljuje točno jednom. Spomenimo kako je nemoguće imati particiju od pet odvojenih skupova s obzirom na to da radimo sa samo četiri vrste karata. Tom problemu možemo doskočiti tako da maknemo jednu kartu pa tako izbacimo i jedno grupiranje ili jednu kartu zamijenimo „joker“ kartom pa tako uvedemo „petu vrstu“. Niz koji tražimo dan je ovako:*

KKKKKTHHHTTKKTTTHTHTPHHPKPPKPPHPKKTHPPTHPHKHPTHPJTKT.

(jedan tref zamijenjen je jokerom)

Definicija 3.3.5. *Neka su* $n, k \in \mathbb{N}$. *Zadan je skup S koji ima n elemenata. Svaki k-člani podskup skupa S zovemo kombinacija k-tog razreda skupa S. Broj svih kombinacija k-tog razreda n-članog skupa označavamo s* $\binom{n}{k}$ *(čitamo "n povrh k"), pri čemu je* $\binom{n}{k} = \frac{n!}{k!(n-k)!}$, *gdje po definiciji uzimamo da je* $0! = 1$.

Primjer 3.3.6. *Kombinacije* 2. *razreda skupa* $\{1, 2, 3, 4, 5\}$ *su:*

$$\{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}, \{4, 5\}.$$

Možemo prebrojati da takvih podskupova ima 10*, što se i poklapa s našom definicijom, tj.* $\binom{5}{2} = 10$.

Pokušajmo pronaći niz duljine 10 koristeći brojeve $1, 2, 3, 4, 5$, svaki po dva puta, tako da se svaki uzastopni par pojavljuje točno jednom (ako je poredak nevažan, tj. $\{1, 2\}$ isto što i $\{2, 1\}$). Kada bismo znali sve o prostim brojevima koji dijele $\binom{2n}{n}$, znali bismo puno više od onoga što matematičari trenutno znaju. Promotrimo primjerice vrijednosti za prvih 10 binomnih koeficijenata $\binom{2n}{n}$ u tablici 3.2.

n	$\binom{2n}{n}$
1	2
2	6
3	20
4	70
5	252
6	924
7	3432
8	12870
9	48620
10	184756

Tablica 3.2: Prvih deset vrijednosti za $\binom{2n}{n}$

Primijetimo da su sve vrijednosti parni brojevi, tj. djeljive su s 2. To očito vrijedi za svaki $n \in \mathbb{N}$ jer za svaki odabir n elemenata iz skupa od $2n$ elemenata postoji njemu komplementaran odabir - stoga je ukupan broj odabira paran. Međutim, vidimo da postoje vrijednosti koje nisu djeljive s 3 (npr. 20 i 70) kao ni s 5 (npr 6 i 252) i 7 (npr. 6, 20 i 3432). Mnogo je teže naći vrijednosti $\binom{2n}{n}$ koje nisu djeljive ni s jednim of 3, 5 i 7. Jedna takva je 2, a iduća je $\binom{20}{10} = 184756 = 2^2 \cdot 11 \cdot 13 \cdot 17 \cdot 19$. Poznati neriješeni problem glasi: "postoji li beskonačan broj takvih n-ova?".

Vratimo se na naš problem: za dani $n, k \in \mathbb{N}$ potrebno je pronaći niz duljine $\binom{n}{k}$ koji sadrži brojeve $\{1, 2, \ldots, n\}$ takve da svaka uzastopna grupa od k elemenata sadrži različitu kombinaciju k-tog razreda skupa $\{1, 2, \ldots, n\}$. Ponekad to možemo napraviti. Primjerice, ako je $n = 8$, $k = 3$, $\binom{8}{3} = 56$, takav niz je dan sa

$$82456145712361246783671345834681258135672568234723578147$$

Teorem 3.3.7. *Za dani $n, k \in \mathbb{N}$ možemo naći niz duljine $\binom{n}{k}$ koji sadrži brojeve iz $S = \{1, \ldots, n\}$ takve da svaka uzastopna grupa od k elemenata sadrži različitu kombinaciju k-tog razreda skupa S: ako k točno dijeli $\binom{n-1}{k-1}$. Obrat ne vrijedi.*

Primjer 3.3.8. *Neka je $n = 4$, $k = 2$. Tada je $\binom{n-1}{k-1} = \binom{3}{1} = 3$ stoga ne možemo naći takav niz. Neka je $n = 8$, $k = 3$, $\binom{8}{3} = 56$, a $3|56$ pa je takav niz dan sa*

$$82456145712361246783671345834681258135672568234723578147.$$

Dokaz. Za $n = 4$ i $k = 2$ postoji 6 kombinacija drugog razreda skupa $\{1, 2, 3, 4\}$. To su: $\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}$. Pretpostavimo da postoji valjani niz duljine šest, označimo ga s $abcdef$. Svaki simbol, a, b, c, d, e ili f označava neki od brojeva 1, 2, 3 ili 4. Znamenka 1 javlja se na nekom mjestu. Na susjednim pozicijama moraju biti neke druge dvije znamenke takve da rezultirajuća dva para predstavljaju neke dvije različite kombinacije drugog razreda koje sadrže 1. 1 se mora pojaviti dva puta u nizu i ta nova jedinica daje još dvije kombinacije koje sadrže 1 no postoji samo tri takve kombinacije 2. razreda što znači da se jedna od kombinacija ponovila što narušava definiciju de Bruijnovog niza stoga ne postoji takav niz. □

Gornji dokaz pokazuje da je „ k dijeli $\binom{n-1}{k-1}$“ nužan uvjet; ako k ne dijeli $\binom{n-1}{k-1}$, onda ne postoji rješenje. Međutim, tvrdnja ne vrijedi u suprotnom smjeru, tj. ne vrijedi: ako k dijeli $\binom{n-1}{k-1}$ onda rješenje mora postojati. Nije teško vidjeti da ciklusi postoje kada je $k = 2$ i n je neparan. Lakše je rukovati s $k = n - 1$ s obzirom na to da $n - 1$ uvijek dijeli $\binom{n-1}{k-1} = n - 1$ i takvi nizovi uvijek postoje. Prvi zanimljiv slučaj je kada je $k = 3$. Brad Jackson je dokazao da ciklusi postoje za sve $n \in \mathbb{N}$ kada je $k = 3$ (i 3 dijeli $\binom{n-1}{2}$).

Njegovi argumenti također su funkcionirali i za $k = 4$. Kasnije je ista stvar dokazana i za $k = 6$ i konačno za $k = 5$. Radi se o teškom i vrlo dugačkom dokazu koji uključuje puno računalnog rada.

Trik koji koristi ova znanja: neka je $k = 3$ i $n = 8$. Znamo da je $\binom{8}{3} = 56$. Ranije dan niz iz primjera 3.3.8 predstavlja niz duljine 56 sastavljen od simbola $\{1, 2, 3, 4, 5, 6, 7, 8\}$ u kojem se svaka tri uzastopna simbola pojavljuju točno jednom. Takav niz ponavlja svaki simbol sedam puta (zbog toga imamo špil od $7 \cdot 8 = 56$ karata).

Poglavlje 4

Programska realizacija de Bruijnovih nizova

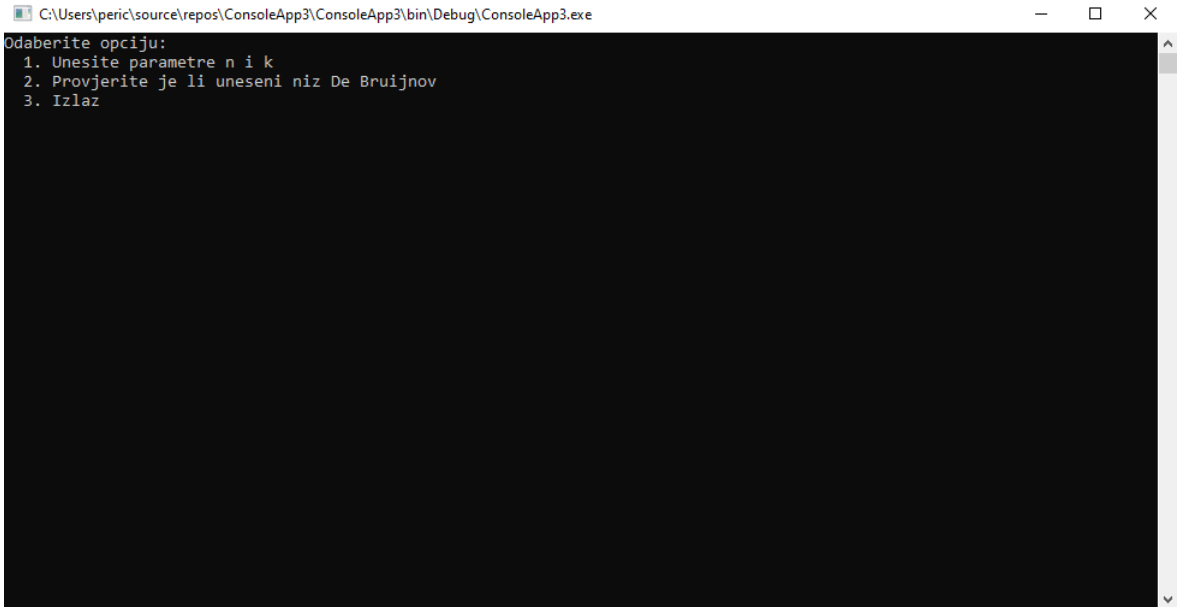
Praktični dio ovog rada je izrada programa za generiranje i provjeru De Bruijnovog niza. Program je rađen u *C#* programskom jeziku kao 'Console application' i ima nekoliko mogućnosti:

- Provjera je li zadani niz de Bruijnov (bez zadavanja parametara $n, k \in \mathbb{N}$, tj. duljine alfabeta i reda niza)
- Ispis svih k-torki za zadane n i k
- Ispis svih vrhova potrebnih za de Bruijnov graf
- Ispih svih veza za Eulerovu šetnju po grafu
- Ispis de Bruijnovih nizova

Na početku samog programa korisnik odabire željenu opciju; provjeru niza ili unošenje parametara n i k. Nakon odabrane opcije korisnik unosi parametre n i k ili niz za provjeru (slika 1).

Dio koda za odabir opcije:

```
int opcija = -1;
while (opcija < 0 || opcija > 3)
{
    Console.WriteLine("Odaberite opciju:");
    Console.WriteLine(" 1. Unesite parametre n i k");
    Console.WriteLine(" 2. Provjerite je li uneseni niz De
        Bruijnov");
    Console.WriteLine(" 3. Izlaz");
    string opcija1String = Console.ReadLine();
    try
    {
        opcija = Int16.Parse(opcija1String);
        if(opcija == 3)
        {
            run = 0;
            Environment.Exit(0);
            continue;
        }
    }
    catch (Exception e)
    {
        opcija = -1;
    }
}
```



Slika 4.1: Početni prozor programa

4.1 Provjera niza

Ako korisnik odabere opciju za provjeru niza, program ga pita za unos niza. Nakon unosa niza, program prvo pronalazi alfabet, te računa n i k. Nakon toga vrši provjeru niza i ispisuje je li niz De Bruijnov ili ne. Ukoliko je, program također ispisuje vrijednosti n i k te alfabet korišten u nizu.

Dohvaćanje alfabeta je jednostvno; program ide "znak po znak" po nizu i sprema znak u listu ukoliko već nije spremljen. Nakon pronalaska alfabeta, lako je izračunati n; to je duljina alfabeta.

```
List<string> alphabet = new List<string>();
foreach(char i in niz)
{
    if(!alphabet.Contains(i.ToString()))
    {
        alphabet.Add(i.ToString());
    }
}
```

```
int n = alphabet.Count;
```

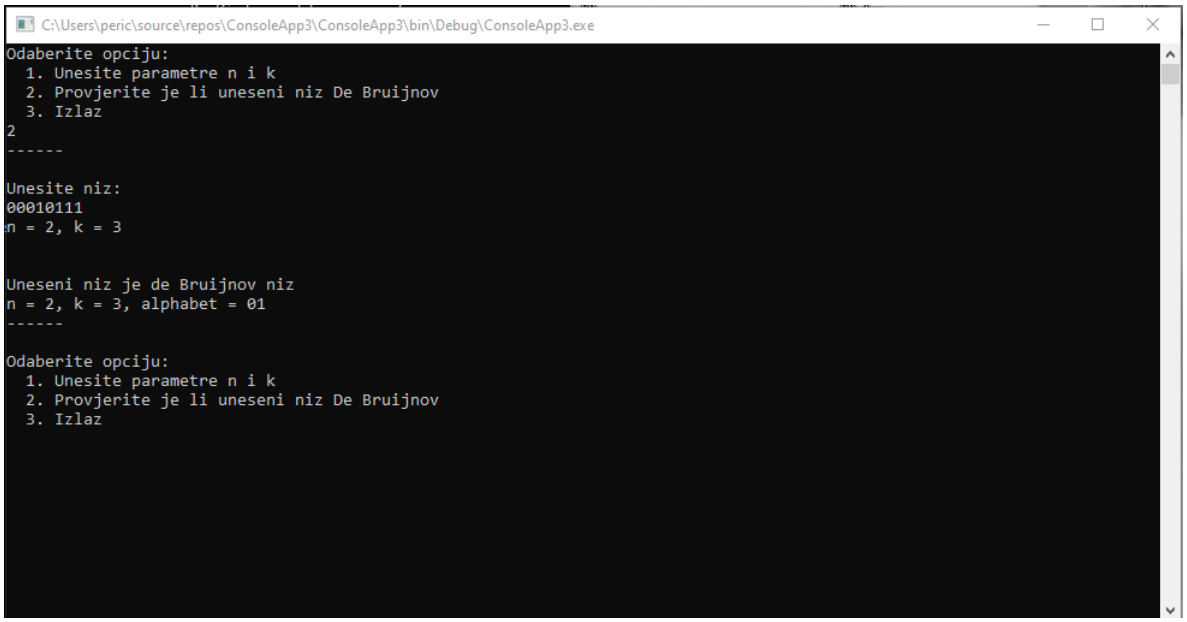
Kad je n poznat, k se može izračunati pomoću logaritma; $log_n(\text{"}duljinaniza\text{"}) = k$, tj. logaritam po bazi n, od duljine cijelog niza (koja je upravo, kao što je poznato n^k) dat će vrijednost za k. Ako k nije cijeli broj, program odmah vraća da niz nije de Bruijnov. Ako je k cijeli broj program ide dalje u provjeru niza.

```
double kTest = Math.Log(niz.Length, n);
if(Math.Floor(kTest) != kTest)
{
    rezultat.isValid = false;
    return rezultat;
}
int k = (int)kTest;
```

Kad je program izračunao n i k, idući korak je dohvaćanje svih k-torki za izračunate vrijednosti n, tj.alfabet i k. Metoda za dohvaćanje svih k-torki će biti obrađena u jednom od sljedećih potpoglavlja. Nakon dobivene listove svih k-torki, program ide po nizu i provjerava pojavljuje li se svaki element iz liste k-torki točno jednom u nizu. Ako se u nizu pojavi k-torka koji nije u listi, ili koja se već prije pojavila, onda niz nije de Bruijnov. Ako lista k-torki na kraju provjere nije prazna, niz također nije de Bruijnov.

```
for(int i = 0; i < niz.Length; i++)
{
    string provjera = "";
    if (i < niz.Length - k)
    {
        provjera = niz.Substring(i, k);
    }
    else
    {
        int ostatak = i + k - niz.Length;
        provjera = niz.Substring(i) + niz.Substring(0, ostatak);
    }
    if (ktorke.Contains(provjera))
    {
        ktorke.Remove(provjera);
    }
    else
    {
```

```
                rezultat.isValid = false;
                return rezultat;
            }
        }
        rezultat.isValid = true;
        return rezultat;
```



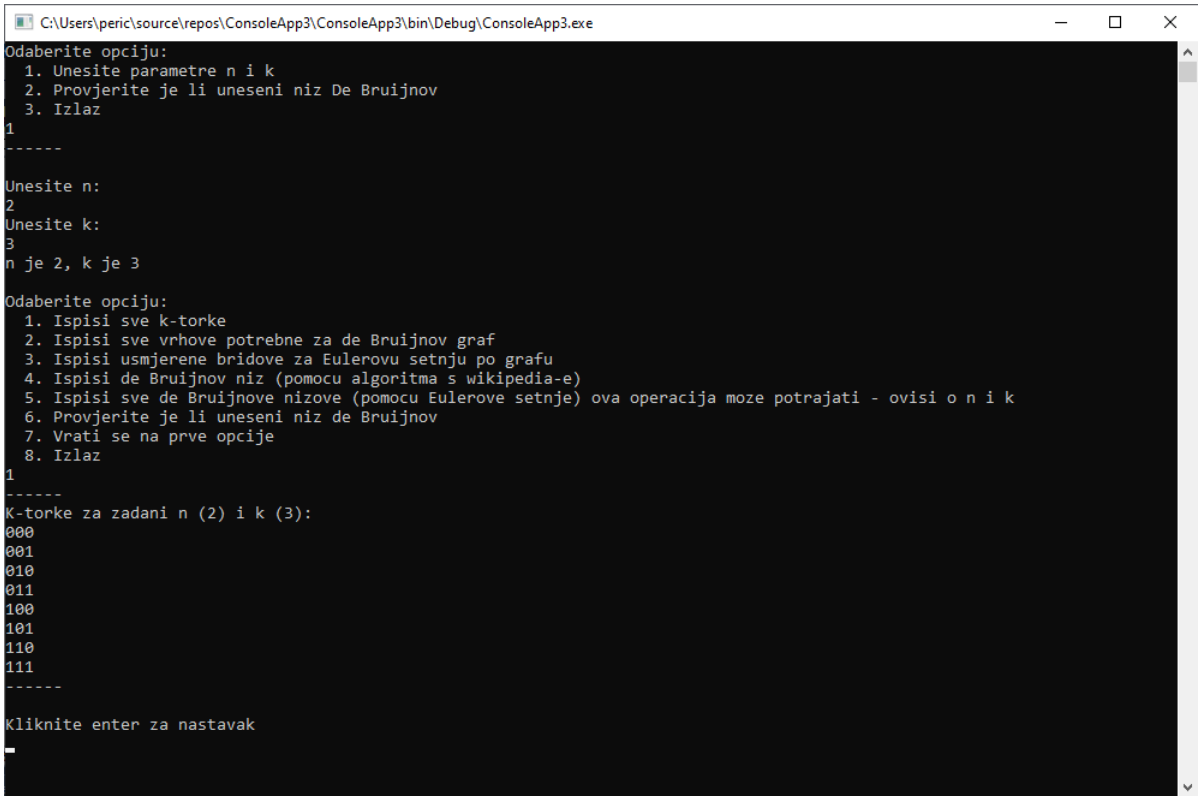
Slika 4.2: Provjera je li niz de Bruijnov

4.2 Ispis svih k-torki

Nakon što korisnik unese n i k, može odabrati opciju ispisivanja svih k-torki. Metoda prima listu u koju će spremiti k-torke, polje svih znamenaka alfabeta i k. Ova se metoda koristi i za ostale izračune, ne samo za opciju prikazivanja k-torki. Nakon primljenih parametara, metoda poziva rekurzivnu metodu koja puni polje k-torkama, smanjuje k za jedan, i ako je k nula, onda je dobivena k-torka duljine k, i ta k-torka se spremu u listu. Navedena operacija ponavlja se n puta; sve dok se ne dobiju sve k-torke za zadane parametre.

```
    static void DohvatiSveKtorke(List<string> ktorke, int[] set, int k)
    {
```

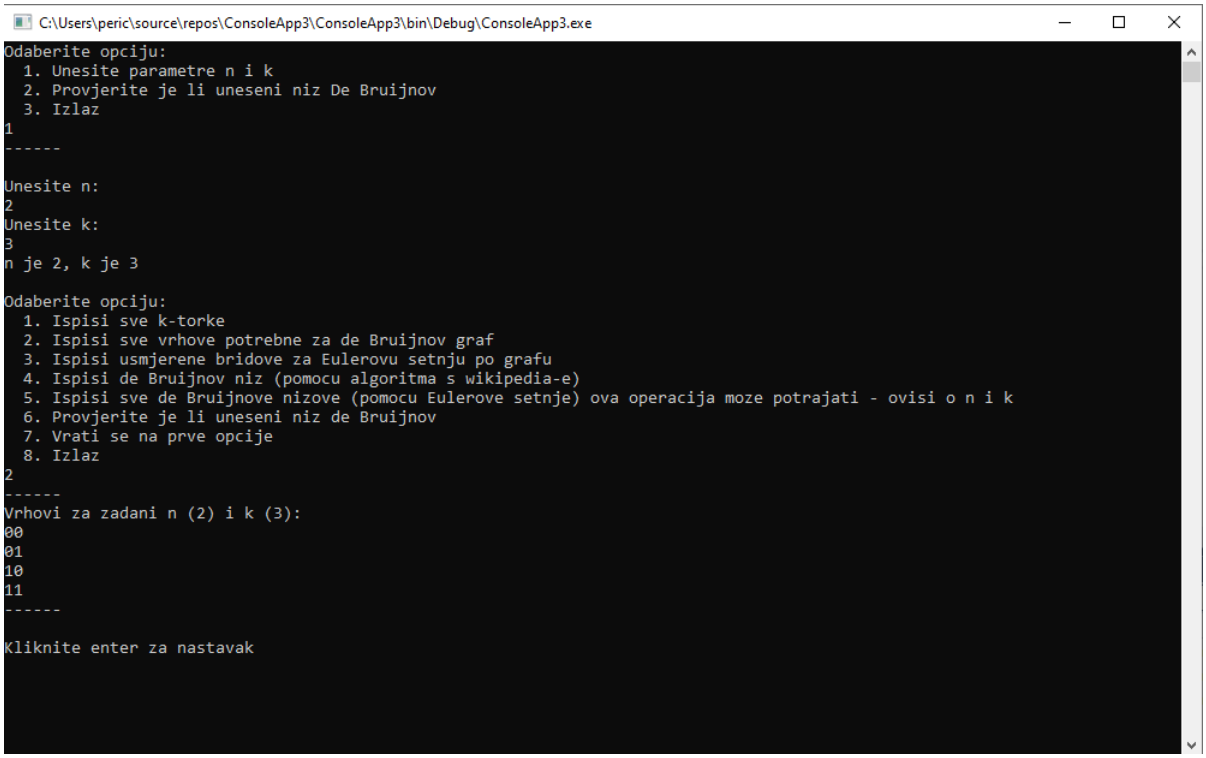
```
        int n = set.Length;
        DohvatiSveKtorkeRekurzija(ktorke, set, "", n, k);
    }

    static void DohvatiSveKtorkeRekurzija(List<string> ktorke, int[]
        set,
                                string prefix,
                                int n, int k)
    {
        if (k == 0)
        {
            ktorke.Add(prefix);
            return;
        }

        for (int i = 0; i < n; ++i)
        {

            string newPrefix = prefix + set[i];

            DohvatiSveKtorkeRekurzija(ktorke, set, newPrefix,
                                n, k - 1);
        }
    }
```



Slika 4.3: Ispis svih k-torki za alfabet duljine n

4.3 Ispis vrhova de Bruijnovog grafa

Ukoliko korisnik želi nacrtati de Bruijnov graf, koristi opciju dohvaćanja svih njegovih vrhova. Ova opcija koristi metodu iz potpoglavlja 4.2, samo umjesto prave vrijednosti k, šalje vrijednost k - 1.



Slika 4.4: Ispis svih (k-1)-torki za alfabet duljine n

4.4 Ispis svih veza za Eulerovu šetnju po grafu

Opcija ispisa svih veza - usmjerenih bridova računa sve veze potrebne za konstrukciju de Bruijnovog grafa i ispisuje ih. Da bi ih izračunao, program prvo dohvaća sve vrhove pomoću metode opisane u prethodnom potpoglavlju. Kad program dobije listu vrhova, sprema sve veze u listu; jedna veza se sastoji od ulaza i izlaza. Program ide kroz dvije petlje po svim vrhovima, i ako je prva znamenka vrha iz prve petlje, jednaka zadnjoj znamenci vrha iz druge petlje, kreira se nova veza, s ulazom vrha iz prve petlje i izlazom vrha iz druge petlje.

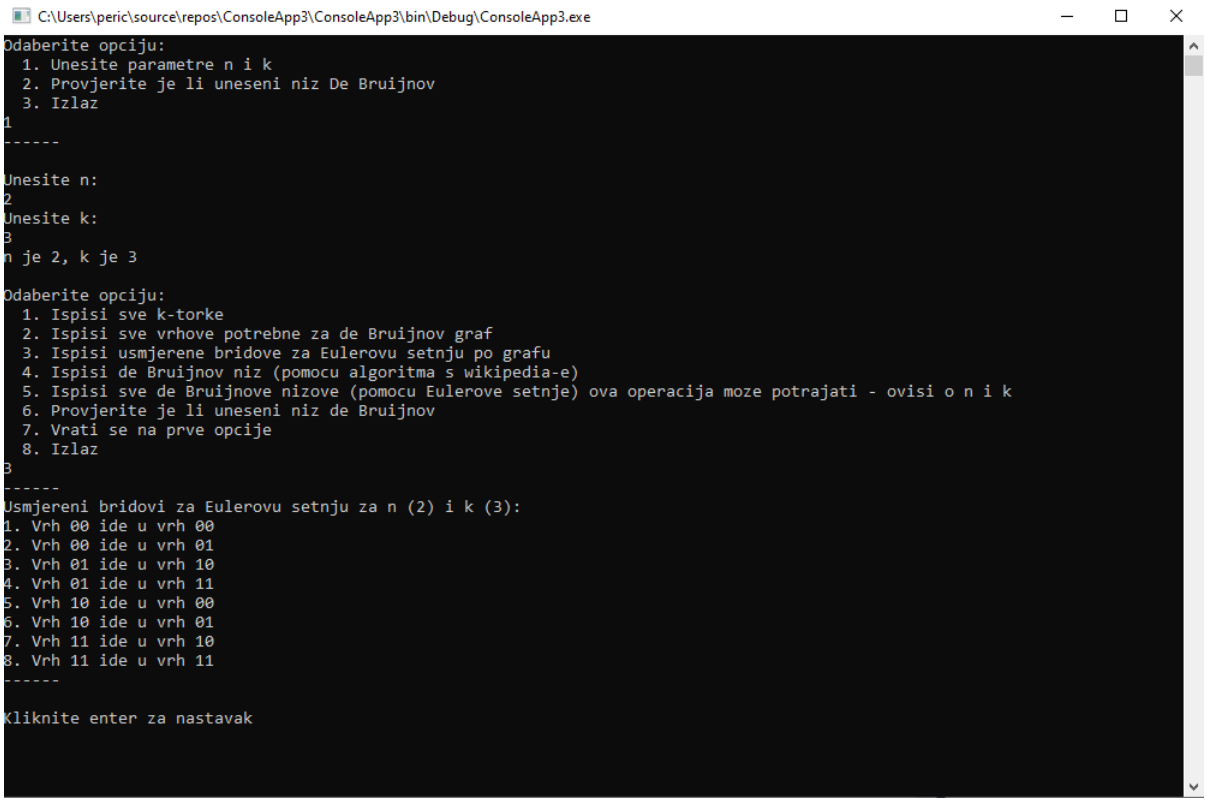
Dio koda koji generira listu veza:

```
private static List<UsmjereniBrid> DohvatiBridove(List<String>
    vrhovi)
```

```
{
    List<UsmjereniBrid> bridovi = new List<UsmjereniBrid>();
    for (int i = 0; i < vrhovi.Count; i++)
    {
        string vrh1 = vrhovi[i];
        for (int j = 0; j < vrhovi.Count; j++)
        {
            string vrh2 = vrhovi[j];
            if (vrh1.Substring(1).Equals(vrh2.Substring(0,
                vrh2.Length - 1)))
            {
                bridovi.Add(new UsmjereniBrid(i,j));
            }
        }
    }
    return bridovi;
}
```

Dio koda koji služi za ispis veza (u korisniku prihvatljivom zapisu):

```
            int i = 1;
            foreach (UsmjereniBrid v in sviBridove)
            {
                Console.WriteLine(i + ". Vrh " +
                    vrhovi.ElementAt(v.ulaz) + " ide u vrh " +
                    vrhovi.ElementAt(v.izlaz));
                i++;
            }
```



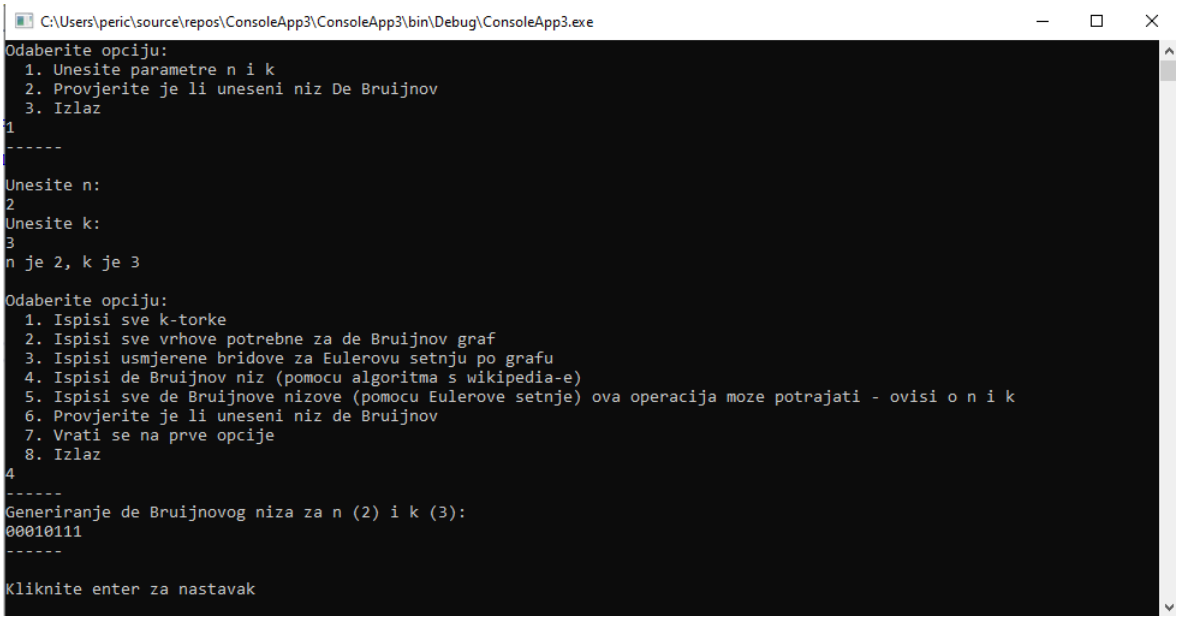
Slika 4.5: Ispis svih veza (usmjerenih bridova) u de Bruijnovom grafu

4.5 Generiranje de Bruijnovog niza

Program generira De Bruijnov niz na dva načina. Prvi koristi algoritam baziran na "Frank Ruskeyevoj Combinatorial Generation" te računa jedan od de Bruijnovih nizova za dane vrijednosti. Ovaj algoritam je puno brži od drugog danog algoritma za računanje niza, ali drugi računa i ispisuje sve nizove za zadane parametre (o čemu će biti više govora u sljedećem potpoglavlju). Kod za algoritam je preuzet s "Wikipedije" te preveden u *C#* stoga je ostavljeno da k predstavlja duljinu alfabeta, a n red de Bruijnovog niza (što je obrnuto od oznaka korištenih u ovom radu).

```
        private static string DeBruijn(int k, int n) //na wikipediji n i k
            su obrnuti parametri
        {
            string alphabet = "";
```

```
        for(int i = 0; i < k; i++)
        {
            alphabet = alphabet + i;
        }
        byte[] a = new byte[k * n];
        List<byte> seq = new List<byte>();

        void db(int t, int p)
        {
            if(t > n)
            {
                if(n % p == 0)
                {
                    seq.AddRange(new ArraySegment<byte>(a, 1, p));
                }
            }
            else
            {
                a[t] = a[t - p];
                db(t + 1, p);
                int j = a[t - p] + 1;
                while (j < k)
                {
                    a[t] = (byte)j;
                    db(t + 1, t);
                    j++;
                }
            }
        }
        db(1, 1);
        StringBuilder sb = new StringBuilder();
        foreach(byte item in seq)
        {
            sb.Append(alphabet[item]);
        }
        string b = sb.ToString();
        return b + b.Substring(0, n - 1);

    }
```



Slika 4.6: Generiranje de Bruijnovog niza za dani n i k

4.6 Generiranje svih de Bruijnovih nizova za zadane parametre n i k

Opcija generira sve de Bruijnove nizove za parametre koje je korisnik unio, međutim, ona je "korisna" za samo mali broj parametara zbog vremena izvršavanja (npr. za $n = 2$ i $k = 3$, postoji 16 različitih de Bruijnovih nizova, a za $n = 2$ i $k = 4$ postoji 256 stoga ti parametri još uvijek nisu problematični no već za $n = 3$ i $k = 3$, postoji $\frac{3!^{3^3}}{3^3}$ različitih de Bruijnovih nizova što je jako velik broj stoga je vrijeme izvršavanja ove operacije jako dugo).

Za računanje svih nizova, prvo se dohvate sve k-torke, potom svi vrhovi i na kraju svi usmjereni bridovi za Eulerov graf. Nakon toga se generira lista 'šetnji'. Svaka šetnja će sadržavati usmjerene bridove sortirane prema načinu na koji je šetnja generirana, te sve usmjerene bridove potrebne za šetnju. Kad se veza(usmjereni brid) doda u sortiranu listu, ista se makne iz liste svih veza. Lista šetnji popunjava se pomoću "for" petlje koja se odvija onoliko puta koliko ima usmjerenih bridova u Eulerovom grafu. Prvi prolaz kroz petlju stvara šetnje za sve usmjerene bridove i dodaje ih u listu. Za idući prolaz kroz petlju, za svaku šetnju se provjera zadnji izlaz iz veze te svi ulazi iz veza kojim odgovara izlaz (iz liste svih veza koja se nalazi u šetnji). Ukoliko postoje četiri takve veze, od postojeće

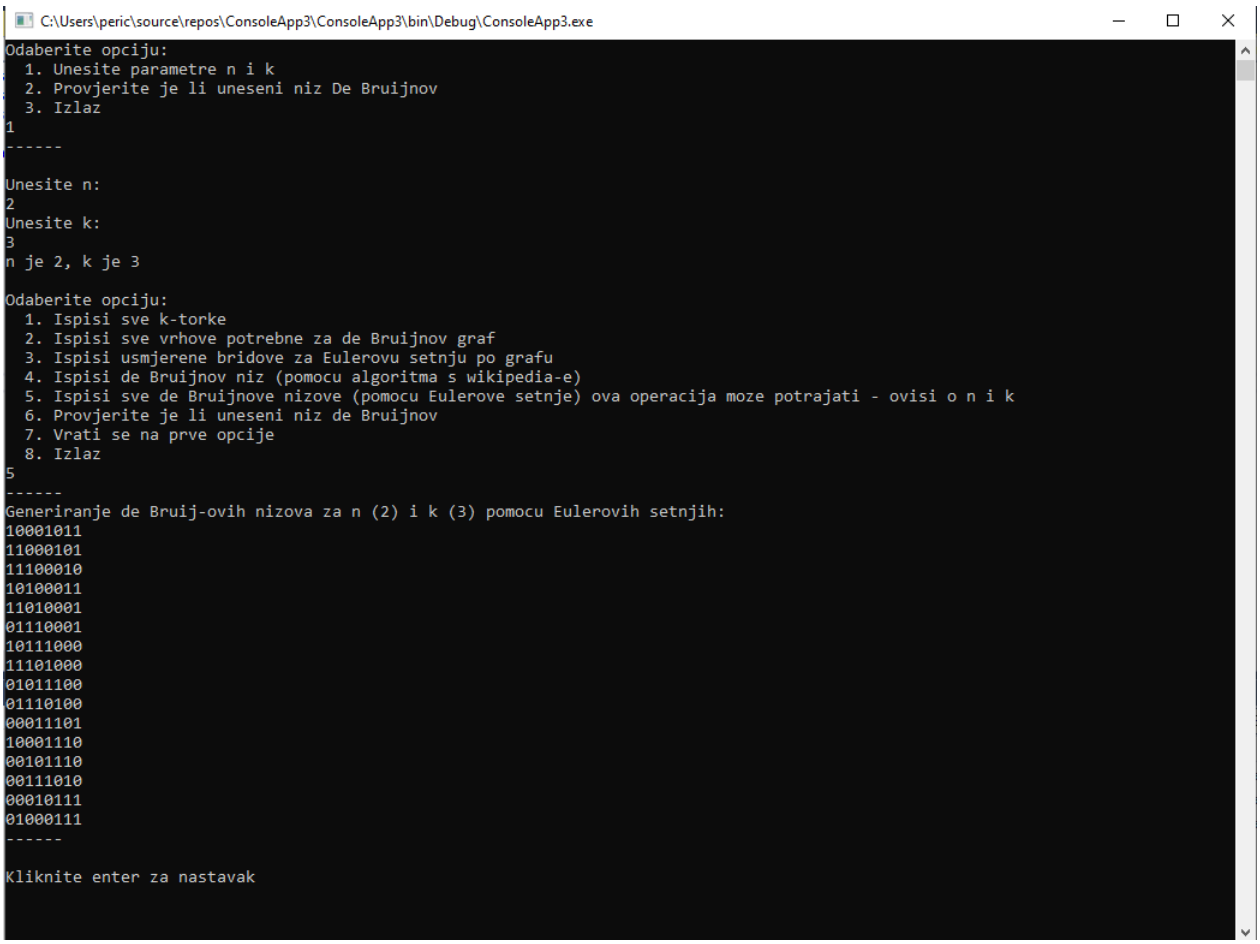
šetnje se naprave još tri šetnje, i u svaku šetnju se doda nova veza. Ako je šetnja došla do 'slijepe ulice' (kad nema ulaza za postojeći izlaz) onda se ona uklanja iz liste.

Nakon generiranja svih šetnji, idući korak je stvaranje De Bruijn-ovog niza od njih. "For" petljom se prolazi kroz sve šetnje i za svaku od njih se generira niz tako da prolazimo kroz sortirane veze koje su spremljene u šetnju. Broj prolaza kroz veze iznosi - duljina svake šetnje umanjena za k^2. Prvi prolaz kroz listu spremi u string cijeli vrh koji odgovara izlazu iz veze (prve veze jer je to prvi prolaz). Svaki idući prolaz dodaje u string samo zadnji znak iz vrha koji odgovara izlazu iz veze.

```
private static List<string> SviNizovi(int n, int k)
       {
           int[] alphabet = new int[(int)n];
           for (int i = 0; i < n; i++)
           {
               alphabet[i] = i;
           }
           List<string> ktorke = new List<string>();
           DohvatiSveKtorke(ktorke, alphabet, k);
           List<string> vrhovi = new List<string>();
           DohvatiSveKtorke(vrhovi, alphabet, k - 1);

           List<UsmjereniBrid> sviBridove = DohvatiBridove(vrhovi);
           List<Setnja> sveSetnje = new List<Setnja>();

           void setnja(int index)
           {
               if(index == 0)
               {
                   foreach(UsmjereniBrid item in sviBridove)
                   {
                       List<UsmjereniBrid> ostaliBridovi = new
                           List<UsmjereniBrid>(sviBridove);
                       ostaliBridovi.Remove(item);
                       Setnja s = new Setnja();
                       s.setnja = new List<UsmjereniBrid>();
                       s.setnja.Add(item);
                       s.bridovi = ostaliBridovi;
                       sveSetnje.Add(s);
                   }
               }
               else
```

```
{
    List<Setnja> tempList = new List<Setnja>(sveSetnje);
    List<Setnja> listaZaBistaSetnje = new List<Setnja>();
    foreach(Setnja s in tempList)
    {
        UsmjereniBrid zadnja = s.setnja.LastOrDefault();
        if(zadnja.izlaz == -1)
        {
            listaZaBistaSetnje.Add(s);
            continue;
        }
        List<UsmjereniBrid> iduciBridovi = new
            List<UsmjereniBrid>();
        foreach (UsmjereniBrid v in s.bridovi)
        {
            if(zadnja.izlaz == v.ulaz)
            {
                iduciBridovi.Add(v);
            }
        }
        if(iduciBridovi.Count == 0)
        {
            s.setnja.Add(new UsmjereniBrid(-1, -1));
            listaZaBistaSetnje.Add(s);
        }
        else if(iduciBridovi.Count == 1)
        {
            s.setnja.Add(iduciBridovi.ElementAt(0));
            s.bridovi.Remove(iduciBridovi.ElementAt(0));
        }
        else
        {
            for(int i = 1; i < iduciBridovi.Count; i++)
            {
                Setnja novaSetnja = new Setnja();
                novaSetnja.setnja = new List<UsmjereniBrid>();
                novaSetnja.setnja.AddRange(s.setnja);
                novaSetnja.bridovi = new List<UsmjereniBrid>();
                novaSetnja.bridovi.AddRange(s.bridovi);
                novaSetnja.setnja.Add(iduciBridovi.ElementAt(i));
                novaSetnja.bridovi.Remove(iduciBridovi.ElementAt(i));
                sveSetnje.Add(novaSetnja);
```

```
                    }
                    s.setnja.Add(iduciBridovi.ElementAt(0));
                    s.bridovi.Remove(iduciBridovi.ElementAt(0));
                }
            }
            foreach (Setnja s in listaZaBistaSetnje)
                sveSetnje.Remove(s);
        }
        index++;
        if(index < sviBridove.Count)
        {
            setnja(index);
        }
    }

    setnja(0);

    List<string> nizovi = new List<string>();
    foreach(Setnja s in sveSetnje)
    {
        StringBuilder sb4 = new StringBuilder();
        for (int i = 0; i < s.setnja.Count - (k - 2); i++)
        {
            UsmjereniBrid v = s.setnja.ElementAt(i);
            if (i == 0)
            {
                sb4.Append(vrhovi.ElementAt(v.izlaz));
            }
            else
            {
                string izlaz = vrhovi.ElementAt(v.izlaz);
                sb4.Append(izlaz.Substring(izlaz.Length - 1));
            }
        }

        nizovi.Add(sb4.ToString());

    }

    return nizovi;

}
```



Slika 4.7: Generiranje svih de Bruijnovih nizova za dani n i k

Bibliografija

[1] Persi Diaconis, Ron Graham, *Magical Mathematics: The Mathematical Ideas That Animate Great Magic Tricks*, Princeton University Press, 2015.

[2] J. Robert Johnson, *Universal cycles for permutations*, (2006), `https://core.ac.uk/download/pdf/82307206.pdf`.

[3] Dujella A., Maretić M., *Kriptografija*, Element, Zagreb, 2007.

[4] Šikić M., Domazet-Lošo M., *Bioinformatika, skripta*, 2013.

[5] Ivica Nakić, *Diskretna matematika, predavanja*, 2011./ 2012.

[6] Wikipedia, *Algoritam za računanje de Bruijnova niza za dani n i k*, `https://en.wikipedia.org/wiki/De_Bruijn_sequence#Algorithm`.

Sažetak

Postoji čitav niz zanimljivih trikova za čije se izvođenje koriste znanja iz matematike. U ovom radu istražuju se trikovi i različite primjene koje se oslanjaju na matematički pojam de Bruijnovih nizova.

Poglavlje 1 započinje kartaškim trikom koji se koristi teorijom grafova, alatima konačnih polja te kombinatorikom [5]. Tu se uvodi pojam de Bruijnovih nizova te je dan odgovor na pitanje koliko de Bruijnovih nizova postoji i opisane su neke metode za konstrukciju de Bruijnovih nizova, a ona koju valja istaknuti jest konstrukcija pomoću de Bruijnovog grafa.

U Poglavlju 2 opisane su primjene de Bruijnovih nizova; načini na koji se oni koriste za robotski vid, u kriptografiji za stvaranje tajnog ključa [3], za rekonstrukciju DNA te računanje udaljenosti između dvije DNA vrpce [4] (gdje se osobito korisnom pokazuje upravo primjena de Bruijnovog grafa).

Poglavlje 3 ponovo se vraća na kartaške trikove čija se izvedba bazira na, primjerice, principu produkta te koji uključuju generalizaciju de Bruijnovih nizova, tj. ono što je definirano kao univerzalni ciklusi.

U poglavlju 4 dana je programska realizacija de Bruijnovih nizova. Program je rađen u programskom jeziku *C#* i ima mogućnosti provjere je li zadani niz de Bruijnov, ispis elemenata potrebnih za konstrukciju de Bruijnovog grafa te ispis de Bruijnovih nizova za dane parametre (duljinu alfabeta n i red niza k).[6].

Summary

There is an entire class of interesting tricks that can be preformed by using math knowledge. In this master thesis, a variety of tricks and different applications relying on de Bruijn sequences are explored.

Chapter 1 starts with a card trick that relies on graph theory, finite fields, and combinatorics[5]. The definition of de Bruijn sequences is given and some methods of construction are described; one of the methodes that should be emphasized is the construction using de Bruijn graph. Also, the answer to the question of the number of de Bruijn sequences is given.

In Chapter 2, applications of de Bruijn sequences are described; in robotic vision, in cryptography for the creation of a corrupting string [3], for DNA reconstruction and for calculating the distance between two DNA strings[4] (for which the use of the de Bruijn graph is remarkably practical).

Chapter 3 goes back to card tricks which are executed based on, for example, the rule of product, and they include the generalization of de Brujin sequences, that is, what is defined as universal cycles.

Chapter 4 gives the program realization of de Bruijn sequences. The program is made in the programming language *C#* and has the ability to check whether the default string is de Bruijn sequence, print the elements needed to construct a de Bruijn graph and print de Bruijn strings for given parameters (length of the alphabet n and order of sequence k)[6].

Životopis

Rođena sam 24. prosinca 1991. u Zagrebu. Živjela sam i odrasla u Novom Zagrebu gdje sam pohađala Osnovnu školu Mladost u naselju Utrina te sam u istom naselju nastavila i srednjoškolsko obrazovanje upisavši Prvu opću gimnaziju koju završavam 2010. Tijekom osnovne i srednje škole aktivno sam se bavila košarkom te igrala i u Prvoj seniorskoj ženskoj ligi. 2010. godine upisujem Prirodoslovno-matematički fakultet u Zagrebu, smjer matematika. Po završetku preddiplomskog studija 2016. godine upisujem Diplomski studij Računarstva i matematike.

U braku sam od 2017. godine i ponosna sam majka trogodišnjoj Antei i jednogodišnjoj Karli.