Rješavanje bugarskog solitera

Brečić, Bože

Undergraduate thesis / Završni rad

2015

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split, University of Split, Faculty of science / Sveučilište u Splitu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:166:401297

Rights / Prava: Attribution-NonCommercial 4.0 International/Imenovanje-Nekomercijalno 4.0 međunarodna

Download date / Datum preuzimanja: **2024-03-28**

Repository / Repozitorij:

Repository of Faculty of Science

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO MATEMATIČKI FAKULTET

ZAVRŠNI RAD

RJEŠAVANJE BUGARSKOG SOLITERA

Bože Brečić

Split, rujan 2015.

Sadržaj

1. Uvod 1

1.1. Povijest bugarskog solitera 1

1.2. Slični problemi 2

1.3. Matematički opis problema 4

1.4. Algoritam za rješavanje problema 7

2. Algoritamsko rješavanje Bugarskog solitera 10

2.1. Razumijevanje Bugarskog solitera 10

2.2. Matematička formulacija Bugarskog solitera 11

2.3. Oblikovanje rješenja Bugarskog solitera 14

2.3.1. Algoritam za rješavanje Bugarskog solitera 14

2.4. Implementacija rješenja Bugarskog solitera u Pythonu 16

2.5. Analiza algoritma 18

3. Zaključak 21

Literatura 22

1. Uvod

Bugarski soliter (engl. Bulgarian solitaire) je matematička kartaška igra koju igra jedna osoba. Igra se sastoji od N karata (ili „novčića“) podijeljenih u nekoliko stupaca. Svaki sljedeći potez se sastoji od oduzimanja po jedne karte iz svakog stupca i od uklonjenih karata formira se novi stupac, pri tome se zanemare stupci s nijednom kartom. Igra je gotova kada se isti poredak ponovi dva puta ili ako je poredak oblika k, (k – 1), (k – 2), ..., 1. Pokazalo se, ako je N triangularan broj, tj. oblika N = k (k + 1) / 2 , za neki pozitivan broj k, da će poredak u stupcima, nakon određenog broja koraka biti k, (k – 1), (k – 2), ..., 1, dok za netrokutaste brojeve N, poredak će se ponoviti.

Cilj ovog rada je opisati matematičke probleme vezane uz Bugarski soliter, te opisati postupak rješavanja problema pronalaskom konačnog poretka.

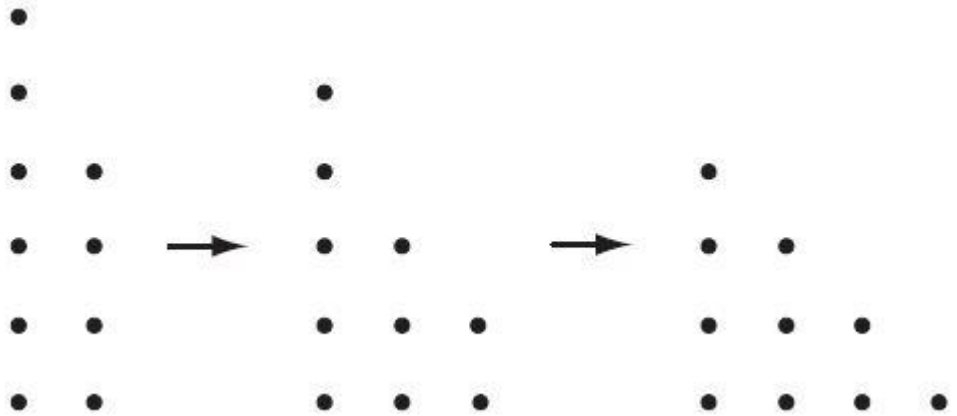
U uvodnom poglavlju se opisuje kratka povijest Bugarskog solitera. Zatim se spominju neke igre koje su usko povezane s Bugarskim soliterom, odnosno nastale se od Bugarskog solitera. Nadalje, iznosi se matematička podloga za ovaj problem, odnosno teoremi koji dokazaju nastajanje karakterističnog ili ponovljenog poretka. Na kraju uvodnog poglavlja teorijski je opisan algoritam koji daje rješenje za uneseni početni problem, te su prikazani primjeri izvršavanja za neke probleme.

U drugom poglavlju se pristupa algoritamskom rješenju problema Bugarskog solitera. Prvo se oblikuje problem, odnosno modeliraju se njegovi ulazni i izlazni parametri i ograničenja. Nakon što se matematički formulira problem, pristupa se oblikovanju algoritma za generiranje i rješavanje problema. Algoritam je implementiran u programskom jeziku Python i njegovo izvršavanje i rezultati izvršavanja se analiziraju nad generiranim problemima.

1.1. Povijest bugarskog solitera

Bugarski solitare je prvi put opisan u ruskom magazinu „Kvant“ 1980. Ima znimljivu priču o nastanku, naime, nepoznati čovjek je u vlaku za Saint Petersbourg ispričao problem ruskom matematičaru Konstantinu Oskolkovu koji je kasnije opisao problem. Moderni naziv za slagalicu imena Bulgarian solitaire dao je Henrik Eriksson s Instituta Tehnology 1982. Matematičar Martin Gardner učinio je igru popularnom kad ju je opisao u časopisu „Scientific

America“ 1983. Mnogi matematičari su opisali ovaj problem, posebno oni koji se bavi teorijom skupova.



Slika 1.1Primjer Bugarskog solitera

Pokazano je da ako je broj trokutast, tj. oblika n = 1 + 2 + ... + k, da će nakon određenog broja koraka biti podijeljen u stupce veličine k, k – 1, k – 2, ... , 1. To znači da ako broj 55 zapišemo u bilo kojem poretku od njegovih 451 276 mogućih, nakon određenog vremenskog intervala da će uvijek završiti sa stupcima veličine 10, 9, 8, ... , 1. Vidimo na slici 1.1 primjer izvršavanja Bugarskog solitera za n = 10, gdje u drugom koraku dolazimo do konačnog poretka. Sergej Popov, brazilski matematičar, je pokazao da čak i ako je isključivanje karata određeno slučajnim odabirom, rezultat će uvijek na kraju biti isti.

1.2. Slični problemi

Postoji nekoliko matematičkih problema koji su ispirirani Bugarskim soliterom. Kao i u orginalnom Bugarskom soliteru, proučavanje problema se odnosi na tip ciklusa i broj karata u pojedinom stogu. Navest ćemo neke od tih sličnih problema.

U Bugarskom soliteru za više igrača (engl. Multiplayer Bulgarian solitaire) na nekoliko particija se događa promjena istovremeno. Svaki stvoreni skup se odmah daje sljedećoj particiji i nikad dva stvorena skupa nisu dana istoj partciji. Ovo možemo opisati kao da više igrača za okruglim stolom igra Bugarski soliter, svaki igrač kad ukloni po jednu kartu iz svakog stupca predaje tu kartu igraču koji sjedi desno od njega. Na slici 1.2 vidimo prikaz koraka za Bugarski soliter s 2 igrača. U svakom koraku oduzimaju po jednu kartu iz svakog sutpca te je predaju drugom igraču koji od tih karata formira novi stupac.



Slika 1.2Prikaz koraka Bugarskog solitera za 2 igrača

Austrijski soliter, moderna igra, prvi put opisana od strane matematičara Ethan Akin i Morton Davis sa sveučilišta u New York-u 1985. Usko je povezana s Bugarskim soliterom. U igri, grupa od N karata je podijeljena u nekoliko stogova. Svaki stog ne može imati više karata od fiksnog broja L. Postoji posebni stog nazvan banka (engl. „bank“). Svaki potez se sastoji od 2 koraka:

- Jedna karta iz svakog stoga se uzme i spremi u stog banka
- Novi stog nastaje iz banke i točno je veličine L, sve dok je veličina banke < L

Igra se nastavlja sve dok se ne ponovi jedinstveni ciklus.

Primjer, uzmimo poredak (5, 4), broj L = 3 i stog banka. Idući korak bit će oduzimanje po jednog broja iz svakog stupca i spremanje u stog banka. Zatim se formira novi stupac, uzimamo L karata iz banke pa će biti (4, 3, 3), a banka sad je sad na -1. U idućem koraku u banku spremamo 3, pa ima vrijednost 2, ponovo formiramo novi stupac veličine L, (3, 2, 2, 3), a banka je ponovo na -1. Tako redom se izvode koraci dok se ne ponovi jedinstveni poredak brojeva ili banka ≥ L.

Montreal soliter je također igra veoma slična Bugarskom soliteru, opisali su je Chris Cannings i John Haigh, matematičari s britanskog sveučilišta, 1990. Igra se sastoji od X karata podijeljenih u nekoliko stupaca, odnosno ima zapis X = (X_m, X_{m-1}, ..., 1), na slici 1.3 je zapis znamenaka u stupcima.

$$\begin{array}{cccccc} m & m-1 & \cdots & 3 & 2 & 1 \\ m+1 & m & \cdots & 4 & 3 & 2 \\ m+2 & m+1 & \cdots & 5 & 4 & 3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m-1+x_m & m-2+x_{m-1} & \cdots & x_3+2 & x_2+1 & x_1 \end{array}$$

Slika 1.3Prikaz znamenaka u Montreal soliteru

U svakom koraku oduzima po jednu kartu iz svakog stupca redom, kao i u Bugarskom soliteru. Razlika je u tome što, kad dođe na prazan stupac, pohrani sve skupljene karte do tada

na to mjesto. Igrač nastavlja radnju sve dok ne dođe do zadnjeg mjesta, time korak završava. Igra je gotova kada se ponovo dođe do početnog položaja. Kao primjer uzmimo poredak (3, 2, 2) → (2, 1, 1, 3) → (1, 0, 0, 2, 4) → (1, 0, 1, 3, 2) → (1, 0, 2, 1, 3) → (1, 1, 0, 2, 3) → ... (nakon 18 poteza) → (3, 2, 2).

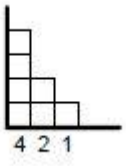
1.3. Matematički opis problema

Prirodni broj može se prikazati kao skupovi, odnosno kao suma cijelih brojeva većih od 0, uključujući sami broj i ne obazirujući se na poredak brojeva u skupu. Tako su skupovi broja 4 : (1 + 1 + 1 + 1), (2 + 1 + 1), (3 + 1) i (4).

λ, lamba znak, koristi se za prikaz skupa prirodnog broja prikazanog kao suma brojeva. U ovom radu λ predstavlja listu koja je zapisana u padajućem poredku, tako da je prvi element ujedno i najveći. Duljina skupa se označava $|\lambda|$.

$\lambda = (\lambda_1 + \lambda_2 + \lambda_3 + ... + \lambda_k : \lambda_1 \geq \lambda_2 \geq \lambda_3 .... \geq \lambda_k : \lambda_1, \lambda_2, \lambda_3, ..., \lambda_k > 0)$, $k = |\lambda|$, gdje je $\lambda_1 + \lambda_2 + \lambda_3 + ... + \lambda_k = N$.

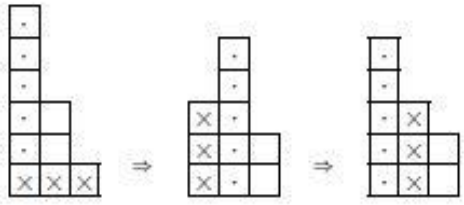
Youngov dijagram služi za vizualizaciju skupova brojeva i poredak karata u Bugarskom soliteru. 7 karata može biti raspoređeno u stupce s 4, 2 i 1 kartom, što može biti opisano kao skup od 7, 4 + 2 + 1 = 7, kao što prikazuje slika 1.2.



Slika 1.4 Youngov dijagram za prikaz skupa broja 7.

Korak u Bugarskom soliteru određen je promjenom u particiji skupa, odnosno ako počnemo s particijom $\lambda = (\lambda_1, \lambda_2, ..., \lambda_k)$ s $\lambda_k > 0$, promjena particije skupa $B(\lambda) = (k, \lambda_1 - 1, \lambda_2 - 1, ..., \lambda_k - 1)$.

Ako je $\lambda_k - 1 > k \geq \lambda_{i+1} - 1$, kao na slici 1.5, onda particiju skupa zapisujemo kao $B(\lambda) = (\lambda_1 - 1, \lambda_2 - 1, ..., \lambda_i - 1, k, \lambda_{i+1} - 1, ..., \lambda_k - 1)$.

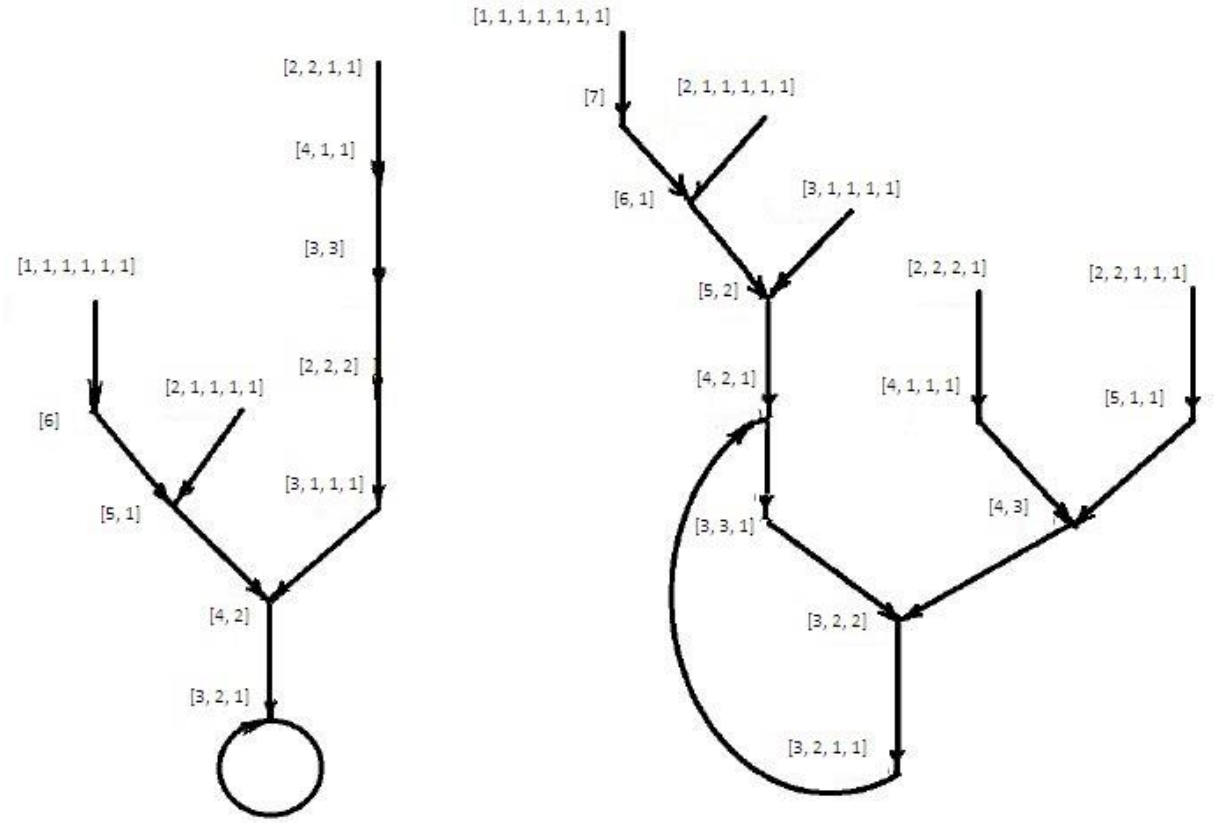


Slika 1.5Prikaz promjene stupaca

Ovo je idealan primjer diskretnog dinamičkog sustava. Mi smatramo P(n) kao skup svih particija broja n i operator B : P(n) → P(n), koji u nekom vremenskom intervalu promjeni stanja sustava, particiju λ , u novo stanje B(λ). Stoga B igra ulogu funkcije ažuriranja sustava. Glavni problem je, počevši od početnog stanje sustava koje označavamo s $\lambda^{(0)}$, utvrditi stanje sustava $\lambda^{(t)} = B(\lambda^{(t-1)})$, koji će doći nakon nekog vremenskog intervala t. Budući da imamo konačan broj stanja P(n), možemo povezati s diskretnim dinamičkim sustavom koji je usmjereni graf s vrhovima particije λ od n, i usmjerenim bridovima (λ, B(λ)).

Trokutasti broj (engl. Triangular number), zapisan kao T_n, je definiran kao 1 + 2 + 3 ...+ k. Zovu ga trokutasti broj jer može formirati trokut, kao 10 čunjeva u kuglanju. U Bugarskom soliteru operacije se ponavljaju na skupu λ. Operacija uključiva umanjivanje svakog broja iz λ za 1, dodajući novi broj u skup koji je jednak $|\lambda|$ i onda uklanjanje 0, ako postoji. Skup nakon i-te operacije je zapisan kao λ^i , što znači da je početni položaj λ^0 , nakon prve operacije skup je označen s λ^1 i tako dalje redom.

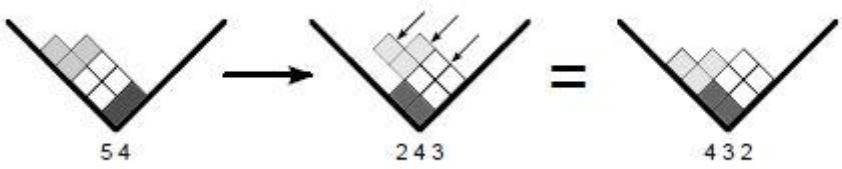
Bugarski soliter je u potpunosti deterministički, specifični skup uvijek vodi specifičnom skupu. Za dani broj karata uvijek postoji konačan broj skupova kojima se taj broj može prikazati, na Slika 1.6 vidimo skupove za n = 6 i n = 7. To znači da se igra može vratiti u prethodni skup. Dakle, Bugarski soliter uvijek završi u ciklusu (duljina ciklusa može biti i duljine 1 kao u skupu 3, 2, 1 → 3, 2, 1). Kada je igra ušla u ciklus, kažemo da je konvergirala.



Slika 1.6 Graf Bugarskog solitera za N = 6 i N = 7

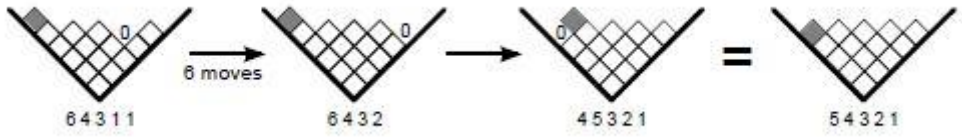
Teorem 1. Kada je ukupan broj karata trokutast broj, Bugarski soliter će se konvergirati u stupce veličine k, k-1, k-2,..., 1.

Dokaz. Za dokaz ćemo iskoristiti vizualni prikaz Andersa Byornera, koji je prikazao Bugarski soliter pomoću Youngovih dijagrama okrenutih za 45 stupnjeva. Svaki put kad se po jedan blok iz svakog stupca uzme, ostali se pomiču u desno stvarajući mjesto novom stupcu. Onda se pusti djelovanje gravitacije da se blokovi slože na pravi način, kao što je prikazano na slici 1.7.



Slika 1.7 Andres Bjornerov dijagram

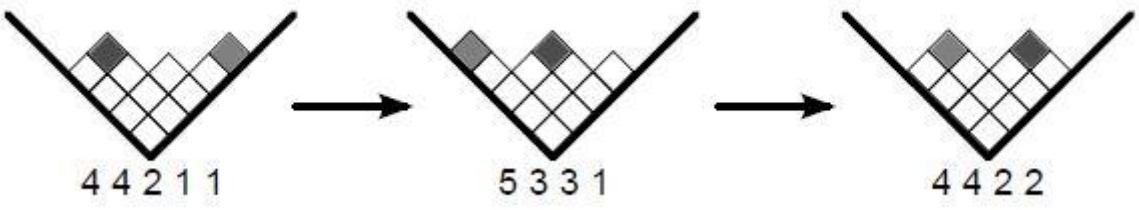
Svaka praznina, u sljedećim koracima, će biti popunjena blokovima. Što znači da će nakon konačnog broja koraka svi blokovi biti na što nižem mogućem položaju, što se postiže kada su blokovi u trokutastom skupu, kao što je primjer za trokutasti broj 15 na slici 1.8.



Slika 1.8 Skraćeni prikaz konvergencije skupa [6, 4, 3, 1, 1]

Teorem 2. Za netrokutaste brojeve, imamo $T_n + r$ blokova, gdje je T_n najveći mogući trokutasti broj. Bugarski soliter će konvergirati u trokutasti skup veličine n na dnu i s viškom r na vrhovima poretka.

Dokaz 2. Uz pomoć Teorema 1 pokazali smo da ispod bloka ne može se pojaviti praznina, odnosno da će je popuniti blok u idućim koracima. Kada nema više praznina za popuniti, na dnu ćemo dobiti trokutasti skup veličine n i još određeni višak blokova, kao što vidimo na Slika 1.9 gdje bijeli blokovi prikazuju trokutasti skup T_n gdje je n = 10, a zatamnjeni blokovi višak r, gdje je r = 2.



Slika 1.9 Dio konvergencije za N = 12.

1.4. Algoritam za rješavanje problema

U matematici, računarstvu i sličnim disciplinama algoritam predstavlja konačan slijed dobro definiranih naredbi za ostvarenje zadatka, koji će za dano početno stanje, nakon konačnog broja koraka vratiti traženo rješenje. Budući da je Bugarski soliter jednostavna igra s kartama ni sam algoritam nije složen. Algoritam dobiva za ulaz listu koja predstavlja skup cijelog broja. Algoritam umanjiva svaki element liste za 1, te dodaje u listu duljinu same liste. Zatim

ukloni sve 0 iz poretka. Ako se ta lista nalazi u dosadašnjem poretku svih listi ili ako trenutna lista zadovoljava konačan poredak koji odgovara trokutasim brojevima, algoritam završava i vraća listu svih poredaka, a ako ne onda se trenutna lista sprema u listu koja sadrži sve poretke i algoritam se vraća na početak.

Pseudokod algoritma koji će rješavati Bugarski soliter izgledao bi:

Algoritam za rješavanje Bugarskog solitera	
1	Function Bugarski_soliter(lista):
2	Postavi sve_liste na []
3	Ponavljaj:
4	Umanji svaki element liste za 1
5	Dodaj u listu duljinu(liste)
6	Ukloni 0 iz liste
7	Dodaj listu u sve_liste
8	Provjeri ponavlja li se poredak
9	Ako je vrati sve_liste
10	Provjeri je li poredak konačan
11	Ako je vrati sve_liste

Algoritam 1 prima za ulaz početni poredak Bugarskog solitera, odnosno niz brojeva i vraća skup lista svih poredaka koji su se dogodili u algoritmu. Nakon što se inicijalno postave prazan skup sve_liste i prazan skup poredak P, ulazi se u petlju koja se izvršava sve dok se ne ponovi poredak ili dok poredak ne bude oblika k, (k – 1), (k – 2), ... , 1. Unutar petlje prođe se kroz svaki element liste te ga se umanji za 1. Zatim se u listu dodaje ukupna duljina liste, odnosno to je ukupan broj oduzetih brojeva. Nakon što se doda u listu ukupan broj oduzetih brojeva, provjerava se je li ta lista, odnosno taj poredak brojeva unutar skupa sve_liste, ako se nalazi vraćamo kao rezultat skup brojeva sve_liste, a ako nije u sve_liste dodajemo i trenutni poredak liste. Provjerava se je li poredak zadovoljavajući, odnosno poredak koji je karakterističan za trokutaste brojeve, ako je taj uvjet zadovoljen ponovno kao rješenje vraćamo skup sve_liste.

U Tablici1 prikazana su rješenja algoritma za zadane početne poretke:

1. Trokutaste: [4, 1, 1] i [8, 4, 3].
2. Netrokutaste: [5, 2, 2] i [7, 7, 2].

Tablica 1Prikaz rješenja za zadane probleme.

[4, 1, 1]	[8, 4, 3]	[5, 2, 2]	[7, 5, 2]
[3, 3]	[7, 3, 3, 2]	[4, 3, 1, 1]	[6, 4, 3, 1]
[2, 2, 2]	[6, 4, 2, 2, 1]	[4, 3, 2]	[5, 4, 3, 2]
[3, 1, 1, 1]	[5, 5, 3, 1, 1]	[3, 3, 2, 1]	[4, 4, 3, 2, 1]
[4, 2]	[5, 4, 4, 2]	[4, 2, 2, 1]	[5, 3, 3, 2, 1]
[3, 2, 1]	[4, 4, 3, 3, 1]	[4, 3, 1, 1]	[5, 4, 2, 2, 1]
	[5, 3, 3, 2, 2]		[5, 4, 3, 1, 1]
	[5, 4, 2, 2, 1, 1]		[5, 4, 3, 2]
	[6, 4, 3, 1, 1]		
	[5, 5, 3, 2]		
	[4, 4, 4, 2, 1]		
	[5, 3, 3, 3, 1]		
	[5, 4, 2, 2, 2]		
	[5, 4, 3, 1, 1, 1]		
	[6, 4, 3, 2]		
	[5, 4, 3, 2, 1]		

Kao što vidimo u Tablici 1, u prvom stupcu za početni poredak (4, 1, 1) nakon 5 koraka smo dobili poredak (3, 2, 1) koji odgovara konačnom poretku za trokutasti broj 6. U drugom stupcu početni poredak je bio (8, 4, 3), algoritam nakon 15-og koraka dolazi do poretka (5, 4, 3, 2, 1) koji odgovara konačnom poretku za broj 15. U trećem i četvrtom stupcu imamo skupove za netrokutaste brojeve 9 i 14, odnosno poretke (5, 2, 2) i (7, 5, 2). U trećem stupcu došlo je do ponavljanja poretka (4, 3, 1, 1) na drugom i petom mjestu, dok je u četvrtom stupcu došlo do ponavljanja poretka (5, 4, 3, 2) na trećem i osmom mjestu.

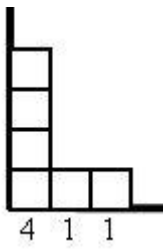
2. Algoritamsko rješavanje Bugarskog solitera

Rješenju Bugarskog solitera se pristupa algoritamski što znači da se prvo pojašnjava problem Bugarskog solitera, odnosno opisuju se ulazni i izlazni parametri problema. Uz pomoć teorije skupova se opisuje problem. Dobivene matematičke formule se koriste za oblikovanje rješenja problema i pri tome se prilagođava algoritam za rješavanje problema Bugarskog solitera. Implementacija algoritma je napravljena u programskom jeziku Python, te su navedene strukture podataka kojima se implementira poredak i specifični koraci algoritma. Na kraju se uspoređuju vremenska i prostorna složenost algoritma implementiranog u Pythonu.

2.1. Razumijevanje Bugarskog solitera

Bugarski soliter je problem koji za dani skup brojeva vraća skup poredaka gdje zadnji poredak označava konačni poredak ili ponovljeni poredak. Brojevi u poretku moraju biti veći od 0.

Ulazni parametar problema je skup brojeva koji označava particiju nekog broja n. Na slici 2.1 imamo skup brojeva (4, 1, 1) koji predstavljaju broj 6.



Slika 2.1 λ = (4, 1, 1)

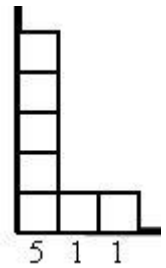
Postoji više skupova kojima se može prikazati broj 6, a također mogu biti ulazni parametri algoritma, prikazani su u tablici 1.

Izlaz iz problema je poredak koji se ponovio ili poredak koji je konačan. Za primjer skupa brojeva sa slike 2.1 bit će skup poredaka: (3, 3), (2, 2, 2), (3, 1, 1, 1), (4, 2), (3, 2, 1) gdje je poredak (3, 2, 1) izlaz iz problema jer se dobio konačni poredak.

6	51	42	411
33	321	3111	222
2211	21111	111111	

Tablica 1 Skupovi broja 6

Za ulazni parametar možemo imati i skup brojeva koji predstavljaju neki netrokutast broj. Na slici 2.3 imamo skup brojeva (5, 1, 1) koji predstavljaju broj 7.



Slika 2.2 $\lambda = (5, 1, 1)$

Ulazni parametar također može biti jedan od 15 skupova koji predstavljaju broj 7, nalaze se u tablici 2.

(7)	(6, 1)	(5, 2)	(5, 1, 1)
(4, 3)	(4, 2, 1)	(4, 1, 1, 1)	(3, 3, 1)
(3, 2, 1, 1)	(3, 1, 1, 1, 1)	(3, 2, 2)	(2, 2, 1, 1, 1)
(2, 2, 2, 1)	(2, 1, 1, 1, 1, 1)	(1, 1, 1, 1, 1, 1, 1)	

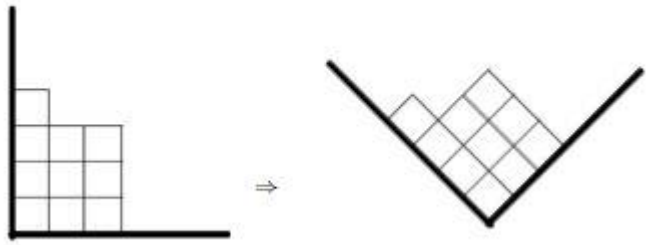
Tablica 2 Skupovi broja 7

Izlazni iz problema za skup brojeva sa slike 2.3 bit će poredak (3, 2, 2) koji se ponavio na drugom i sedmom mjestu u rješenju. Skup rješenja bit će : (4, 3), (3, 2, 2), (3, 2, 1, 1), (4, 2, 1), (3, 3, 1), (3, 2, 2).

2.2. Matematička formulacija Bugarskog solitera

Ulazni parametar za Bugarski soliter se može shvatiti kao skup brojeva čija suma daje neki broj n, odnosno $\lambda = (\lambda_1, \lambda_2, ..., \lambda_k)$, $\lambda_k > 0$ gdje je $n = \lambda_1 + \lambda_2 + ... + \lambda_k$. Iduća particija skupa će biti $B(\lambda) = (k, \lambda_1 - 1, \lambda_2 - 1, ..., \lambda_k - 1)$. Ako je $\lambda_k - 1 > k \geq \lambda_{i+1} - 1$, onda particiju skupa zapisujemo kao $B(\lambda) = (\lambda_1 - 1, \lambda_2 - 1, ..., \lambda_i - 1, k, \lambda_{i+1} - 1, ..., \lambda_k - 1)$. Iskoristimo Andres Bjornerov dijagram za prikaz Bugarskog solitera. Ako okrenemo Youngov dijagram za 45

stupnjeva suprotno od kazaljke na satu dobit ćemo dijagram za vizualizaciju poretka kao na slici 2.5.

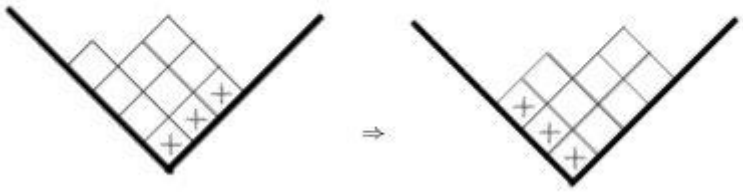


Slika 2.3 Youngov i Andres Bjornerov dijagram za (4, 3, 3)

Uz pretpostavku da su sve kutije iz poretka λ materijalne točke jednake mase m, možemo razmatrati potencijalnu energiju sustava kao:

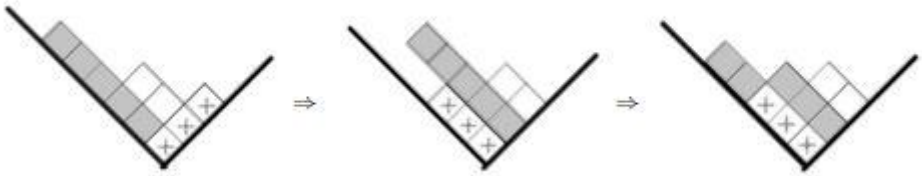
$$U(\lambda) = mg \sum \text{hij}$$

Gdje je g ≈ 9, 81 m / s^2 akceleracija slobodnog pada na zemlji, suma je jednaka za svaku kutiju iz poretka λ, a h_{ij} je visina središta kutije s koordinatama (i, j), gdje j odgovara broju kutije u i-tom poretku. Kao što smo ranije rekli, korak Bugarskog solitera sastoji se od uklanjanja k kutija iz donjeg retka i postavlja ih kao prvi stupac, kao što je prikazano na slici 2.4. Na ovaj način kutija (1, j) ∈ λ postaje kutija (j, 1) ∈ B(λ). Potencijalna energija tih k kutija se ne mijenja. Ostale n – k kutije se iz poretka λ se pomiču jedan korak udesno, iz pozicije (i, j), j $>$ 0 na poziciju (i + 1, j – 1). Stoga će oni sačuvati svoju potencijalnu energiju. Dakle, ako je k $\geq$ λ_1 – 1, kao na slici 2.6, korak Bugarskog solitera prisiljava kutije u ciklusu na istoj razini i čuva potencijanu energiju dijagrama.



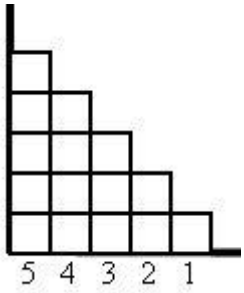
Slika 2.4 B(λ) = (4, 3, 3) → B(λ) = (3, 3, 2, 2)

Ako je $k \leq \lambda_1 - 1$, kao na slici 2.7, onda će gravitacija privući kutije koje su višak u drugom stupcu i one će pasti, a potencijalna energija dijagrama će se smanjiti.



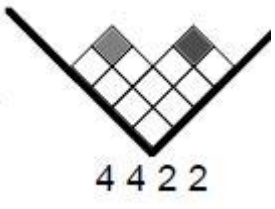
Slika 2.5 $B(\lambda) = (6, 3, 1) \rightarrow (3, 5, 2) \rightarrow (5, 3, 2)$

Ako je broj n oblika $n = k (k + 1) / 2$, onda će nakon nekog vremenskog intervala konvergirati u stabilni poredak $(k, (k – 1), (k – 2), ..., 1)$. Na slici 2.5 je prikazan stabilan poredak za n = 15, gdje k iznosi 5.



Slika 2.6Prikaz trokutastog broja 15

U suprotnome, poredak će biti oblika Tn + r, gdje je Tn najveći mogući trokutast broj za broj n, a r predstavlja višak. Kao što vidimo na slici, kvadrati ispunjeni bijelom bojom prikazuju Tn = 10, a kvadratići ispunjeni sivom bojom višak r = 2, za broj n = 12.



Slika 2.7 Prikaz netrokutastog broja 12

2.3. Oblikovanje rješenja Bugarskog solitera

Rješenje Bugarskog solitera je algoritam koji za zadani početni poredak skupa brojeva vraća sve poretke koji se dogode dok se ne dođe do konačnog ili ponovljenog poretka. Pošto je igra dosta jednostavna, nije potreban nijedan algoritam pretraživanja, nego sam jedan deterministički algoritam koji generira poretke sve dok ne dođe do konačnog ili ponovljenog poretka.

2.3.1. Algoritam za rješavanje Bugarskog solitera

Algoritam za rješavanje Bugarskog solitera izvršava određene korake nad početnim poretkom kojeg dobije kao ulazni parametar. Nakon što umanji za jedan svaki element iz poretka, u listu dodaje ukupan broj oduzetih brojeva. Kako u poretke zanemarivamo prazne stupce, provjerava se svaki element liste poretka te ako je neki od njih jednak 0, on se ukloni iz poretka. Trenutni poredak se sprema u skup rješenja. Nadalje, provjera vas se je li taj poredak već u listi rješenja, te je li taj poredak konačan. Ako je zadovoljen jedan od ta dva uvjeta, algoritam se prekida i kao izlaz se šalje skup rješenja, odnosno skup svih poredaka, gdje zadnji poredak predstavlja konačan ili ponovljen poredak.

Algoritam 1 Rješavanje Bugarskog solitera

```
1.  Function Bulgarian_soliter(lista):
2.    sve_liste ← []
3.    P ← []
4.    do:
5.      for element ∈ lista:
6.        element ← element - 1
7.      lista ← lista + [|lista|]
8.      for element ∈ lista:
9.        if element != 0
10.         P ←P + lista
11.     P ← ReverzniSort(P)
12.     if P ∈ sve_liste:
13.       return sve_liste
14.     else:
15.       sve_liste ← sve_liste + [P]
16.     X ← 0
17.     P  ← sort(P)
18.     for elem od 0 do |P|:
19.       if P[elem] = elem + 1
20.         X ← X + 1
21.     if X = |P|:
22.       return sve_liste
```

Ako pogledamo izvršavanje algoritma 1 :

1. Algoritam kao ulazni parametar dobiva listu, koja predstavlja početni poredak
2. Lista rješenja sve_liste i poredak P se postavljaju na prazne liste
3. Dok se ne dođe do ponovljenog ili konačnog poretka algoritam će se izvršavati
4. Prolazi se kroz svaki element liste te ga se umanji za 1
5. U listu se dodaje ukupan broj elemenata liste, što odgovara broju oduzetih brojeva
6. Prolazi se kroz svaki element liste te se uklanjaju oni elementi koji su jedanki 0
7. Lista se sprema u poredak P

8. Sortiramo silazno elemenate u poretku P
9. Ako se poredak P nalazi u rješenju sve_liste, vrati sve_liste
10. U suprotnom, u sve_liste dodaj poredak P
11. Poredak P sortiramo uzlazno
12. Prolazi se kroz svaki element liste, ako je svaka vrijednost elementa jednaka indeksu elementa + 1 onda je poredak konačan, vrati sve_liste

Za pokazati izvršavanje algoritma 1 uzet ćemo početni poredak sa slike 2.1, gdje je poredak jednak (4, 1, 1). U tablici 3 vidimo prikaz promjena varijabli u svakom koraku.

Tablica 3 Prikaz koraka za (4, 1, 1)

Varijable / Koraci	lista	P	sve_liste
Korak 1	[4, 1, 1]→[3, 0, 0] →[3, 0, 0, 3] →[3, 3]	[3, 3]	[3, 3]
Korak 2	[3, 3] →[2, 2] →[2, 2, 2]	[2, 2, 2]	[[3, 3], [2, 2, 2]]
Korak 3	[2, 2, 2] →[1, 1, 1] →[1, 1, 1, 3]	[3, 1, 1, 1]	[[3, 3], [2, 2, 2], [3, 1, 1, 1]]
Korak 4	[3, 1, 1, 1]→[2, 0, 0, 0]→[2, 0, 0, 0, 4] →[2, 4]	[4, 2]	[[3, 3], [2, 2, 2], [3, 1, 1, 1], [4, 2]]
Korak 5	[4, 2] →[3, 1] →[3, 1, 2]	[3, 2, 1]	[[3, 3], [2, 2, 2], [3, 1, 1, 1], [4, 2], [3, 2, 1]]

2.4. Implementacija rješenja Bugarskog solitera u Pythonu

Algoritam za rješavanje Bugarskog solitera implementiran je u programskom jeziku Python. Struktura podataka koja se koristi za ulazni parametar, odnosno početni poredak, je lista brojeva, čija vrijednost svakog elementa predstava broj kutija u stupcima. Ako početni poredak ima u prvom stupcu 6 kutija, u drugom stupcu 4 kutije, u trećem stupcu 2 kutije, onda će poečtni poredak biti 6, 4, 2, odnosno algoritam će dobiti kao ulazni parametar listu brojeva (6, 4, 3). Struktura podataka koja se koristi za izlazni parametar, odnosno skup poredaka, biti će lista u listi. Rješenje će sadržavati skup lista, gdje zadnja lista predstavlja konačni ili

ponovljeni poredak. Za ulazni parametar (6, 4, 2) vratit će [[5, 3, 3, 1], [4, 4, 2, 2], [4, 3, 3, 1, 1], [5, 3, 2, 2], [4, 4, 2, 1, 1], [5, 3, 3, 1]], gdje je poredak [5, 3, 3, 1] ponovljeni poredak.

```
1   def Bulgarian_Solitaire(lista):
2       sve_liste=[]
3       poredak=[]
4       while(True):
5           niz=[]
6           for i in range(len(lista)):
7               lista[i] -= 1
8           lista.append(len(lista))
9           for i in range(len(lista)):
10              if (lista[i] != 0):
11                  niz.append(lista[i])
12          lista=sorted(niz, reverse = True)
13          if(lista in sve_liste):
14              sve_liste.append(list(lista))
15              return sve_liste
16          else:
17              sve_liste.append(list(lista))
18          poredak = sorted(lista)
19          x=0
20          for i in range(len(poredak)):
21              if(poredak[i] == i+1):
22                  x = x+1
23              if x == len(lista):
24                  return sve_liste
```

Kôd 2.1 Implementacija algoritma za rješavanje Bugarskog solitera

Implementacija algoritma za rješavanje Bugarskog solitera 2.1 u Pythonu veoma je slična pseudokodu algoritma 1. Funkcija Bulgarian_solitaire() za parametar prima početni poredak, odnosno listu brojeva koji predstavljaju taj poredak. U drugoj i trećoj liniji koda liste sve_liste i poredak se postavljaju na prazne liste. Nadalje, u 4-oj liniji se ulazi u petlju koja će se izvršavati sve dok se ne dođe do ponovljenog ili konačnog poretka. Postavljamo lokalnu varijablu niz na praznu listu, koja će služiti za čuvanje poretka. U 6-oj liniji koda ulazimo u petlju koja prolazi kroz svaki element u listi, te postavlja vrijednost svakom elementu umanjenu za 1. Zatim u listu dodajemo duljinu same liste, jer upravo je to ukupan broj

oduzetih brojeva. Petlja u 9-oj liniji koda u lokalnu varijablu niz sprema sve elemente iz liste koji su različiti od 0. Zatim u listu spremimo vrijednost varijable niz sortiran silazno. U 13-oj liniji koda provjeravamo nalazi li se lista u listi rješenja sve_liste, ako nalazi u sve_liste spremamo listu i vraćamo rezultat sve_liste, a ako ne nalazi onda samo spremimo listu u sve_liste i nastavljamo dalje. U poredak spremimo listu sortiranu uzlazno, da bi mogli provjeriti je li poredak u listi konačni poredak koji odgovara Bugarskom soliteru. Potrebna nam je još jedna lokalna varijabla x koju postavljamo na vrijednost 0. U 20-oj liniji ulazimo u petlju koja ide od 0 do duljine poretka. Provjeravamo je li element poretka jednak indeksu poretka + 1, ako je, inkrementiramo varijablu x. Ako je poredak konačan, odnosno da je za svaki element njegova vrijednost jednaka vrijednosti njegova indeksu + 1 onda bi u varijabli x trebala biti vrijednost jednaka duljini poretka. Ako je to istinito vraćamo rezultat sve_liste, u suprotnom idemo na idući korak u Bugarskom soliteru.

Implementirana je još i testna funkcija koju vidimo u Kôdu 2.2. Ona za ulazni parametar prima početni poredak, koji je predstavljen strukturom podataka lista, te ocekivani izlaz koji je predstavljen strukturom podataka lista u listi, koji predstavljaju rješenje za početni poredak. Unutar funckije test() se provjerava je li ocekivani izlaz jednak rješenju dobivenom pomoću funckije Bulgarian_solitaire kojoj se šalje taj početni poredak te se ispiše odgovor.

```
1  def test(ulaz, ocekivani_izlaz):
2      dobiveni_izlaz = Bulgarian_Solitaire(ulaz)
3      print(ocekivani_izlaz == dobiveni_izlaz)
4
```

Kod 2 Implementacija testne funckcije

2.5. Analiza algoritma

Algoritam za rješavanje Bugarskog solitera nije složem, stoga i samo izvršavanje ne traje vremenski dugo. U potpunosti je deterministički, odnosno koji će pri svakom izvršavanju u bilo kojim uvjetima od istog unosa doći do istog izlaza, sljedeći svaki put jednak niz naredbi

U tablici 4 su prikazani vremenski intervali potrebni da se algoritam izvrši za pojedine brojeve n, odnosno skupove brojeva kojima se prikazuje taj broj n, te u stupcu prostor prikazan je broj bajtova (engl. bytes) koji prikazuju iskorištenje memorije.

Vidimo da vrijeme izvršavanja algoritma raste s porastom broja n. Ovisi i kojim poretkom je prikazan dani broj n. Stoga vidimo da su skupovi (21, 13, 11, 2) i (25, 11, 9, 2) particije broja 47, a imaju različito vrijeme izvršavanja. Kako smo ranije spomenuli, particije nekog broja možemo prikazati pomoću usmjerenog grafa, Slika 1.6, stoga ovisi koliko koraka, odnosno vrhova, treba prijeći od početnog do konačnog poretka. Stoga, dva poretka za neki broj n mogu imati različito vrijeme izvršavanja, kao što vidimo u Tablica 5 za poretke broja 6.

Broj N, poredak za N	Vrijeme	Prostor
N = 14, (10, 2, 2)	0.00011	100
N = 29, (17, 5, 3, 4)	0.000990	176
N = 47, (21, 13, 11, 2)	0.000996	220
N = 47, (25, 11, 9, 2)	0.000999	236
N = 163, (125, 21, 9, 8)	0.003001	540
N = 265, (125, 121, 19)	0.004020	628
N = 459, (235, 115, 109)	0.012011	1452
N = 605 (335, 115, 100, 55)	0.159999	2428

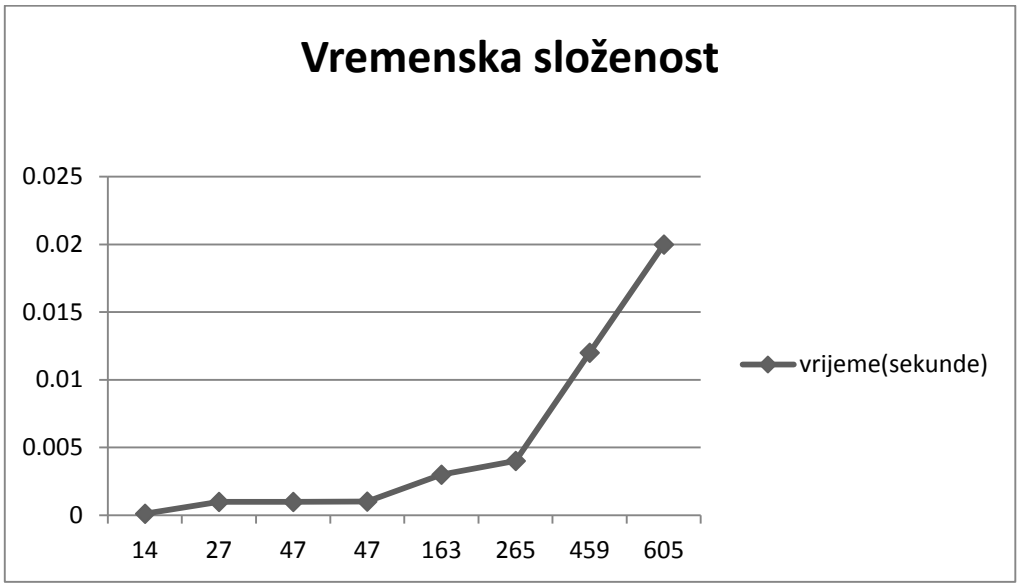
Tablica 4 Prostor i vrijeme za izvršavanje algoritma

Poredak za N = 6	Vrijeme	Prostor
(6)	$1.000165 * 10^3$	52
(5, 1)	$0.999992 * 10^3$	52
(4, 2)	$0.999902 * 10^3$	52
(3, 1, 1, 1)	$0.999992 * 10^3$	52
(3, 3)	$1.000299 * 10^3$	52
(2, 2, 2)	$1.000165 * 10^3$	52
(2, 2, 1, 1)	$1.000441 * 10^3$	52
(2, 1, 1, 1, 1)	$1.000165 * 10^3$	52
(1, 1, 1, 1, 1, 1)	$1.000299 * 10^3$	52
(3, 2, 1)	$0.990016 * 10^3$	52

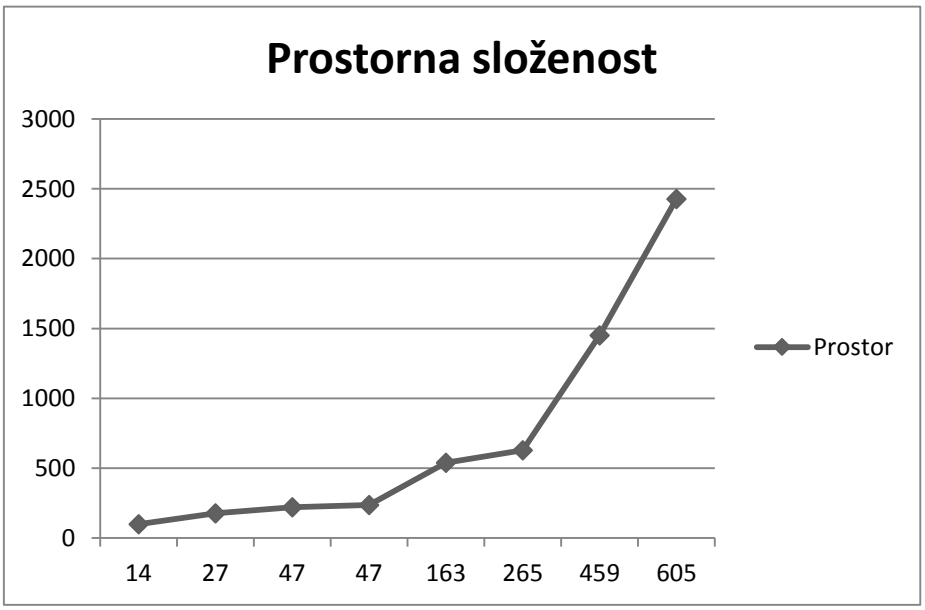
Tablica 5 Vrijeme i prostor za izvršavanje algoritma za poretke broja 6

Na grafikonu 1 vidimo kako raste vremenska složenost eksponencijalno s porastom broja N. Kako sam algoritam nije zahtjevan nije potrebno ni puno vremena za izvršiti, ali ipak uočavamo promjenu. Vremenska složenost također ovisi i o poretku brojeva u listi, ne samo veličini broja N.

Grafikon 2 prikaziva prostornu složenost algoritma te vidimo da i ona raste eksponencijalno kako se povećava broj N.



Grafikon 1 Vremenska složenost algoritma



Grafikon 2 Prostona složenost algoritma

3. Zaključak

Bugarski soliter je matematička igra koju je opisao Konstantin Oskolkov. Privlači pažnju mnogih matematičara iz raznih područja, najviše teorije skupova, diskretne matematike, itd. Postoje razni teoremi i dokazi o postojanju stabilnog poretka za trokutaste brojeve i ponavljanje poretka za netrokutaste brojeve. Ostaje još za dokazati formulu za najveći broj poteza prije konvergencije u stabilni poredak. Koliko znamo, nitko nije uspio dokazati takvu formulu koja vrijedi za sve brojeve. Vidjeli smo u radu da su mnoge igre nastale iz ovog problema. Problem je što je igra predvidljiva, možda bi bila zanimljivija kad bi imala više pravila.

U ovom radu opisan je postupak rješavanja problema Bugarskog solitera za bilo koji dani početni poredak koji zadovoljava ograničenja. Oblikovan je algoritam koji za ulaz dobije listu, koja predstavlja poredak, a vrati listu svih poredaka koji su se dogodili prije konačnog ili ponovljenog poretka. Algoritam je implementiran u programskom jeziku Python, te su rezultati za pojedine početne probleme uspoređivani i pokazano je da vremenska i prostorna složenost raste eksponencijalno s porastom broja N.

Literatura

[1] E. Atkin, M. Davis, »Bulgarian solitaire« Amer. Math. Montly 92, pp. 237-250, 1985.

[2] S. Popov, »Random Bulgarian solitaire« Random structures and algorithms, pp. 310-330, 2005.

[3] Wikipedia, » Bulgarian solitaire« 2015. https://en.wikipedia.org/wiki/Bulgarian_solitaire

[4] A. Grensjo, »Bulgarian solitaire in Three Dimensions« Royal Institute of Technology, Stockholm, 2013.

[5] H. Eriksen, »Multiplayer Bulgarian solitaire« Royal Institute of Technology, 2012.

[6] B. Bojanov, »The Bulgarian solitaire and the mathematics around it« 2015.

[7] Mancala world, »Austrian solitaire« 2015.
http://mancala.wikia.com/wiki/Austrian_Solitaire

[8] A. Levitin, »Algorithmic puzzles«, Oxford University, 2011. pp. 238-241.

Rješavanje Bugarskog solitera

Sažetak

Poopćenje Bugarskog solitera je problem pronalaska svih poredaka brojeva koji se dogode dok poredak ne bude konačan za trokutaste brojeve ili dok se poredak ne ponovi. Mnogi matematički problemi se mogu reducirati na problem Bugarskog solitera. Istraženo je rješenje koje heurstičkim pristupom rješava problem. Prikazan je postupak optimalnog rješavanja poopćenog problema Bugarskog solitera algoritmom. Algoritam je implementiran u Pythonu. Rezultati izvršavanja algoritma su analizirani i uspoređeni, te je pokazana vremenska i prostorna složenost za algoritam koji rješava Bugarski soliter.

Ključne riječi: Bugarski soliter, trokutasti brojevi, algoritam

Solving Bulgarian solitaire

Abstract

Generalization of Bulgarian solitaire is problem of finding every position of number that occur while position is not in the stable position for triangular number or until position is not repeated. Many mathematical problems can be reduced by Bulgarian solitaire. Solution that use heuristics to solve problem are researched. Procedure of optimal solving generalized Bulgarian solitaire using algorithm is shown. Algorithm are implemented in Python. Results of execution algorithm are analyzed and compared, time and space complexity for algorithm are shown.

Keywords: Bulgarian solitaire, triangular number, algorithm