Gimnazija Ledina

Resljeva 12, 1000 Ljubljana

Zlate palice

Raziskovalna naloga na področju računalništva

Avtor: Jon Novak, 4. letnik

Mentor: Betka Burger

Ljubljana, 2024

Zahvala

Rad bi se zahvalil svoji mentorici Betki Burger za pomoč pri izdelavi raziskovalne naloge. Zahvaljujem se tudi svojemu profesorju matematike Urošu Cotmanu, da me je navdušil za raziskovanje matematičnega ozadja logičnih iger. Prav tako bi se rad zahvalil tudi Mateju Korbarju za pomoč pri integriranju programske kode v spletno stran. Posebna zahvala pa gre mojemu prijatelju Lovru Basarju za vso pomoč pri izdelavi spletne strani.

Kazalo

Kazalo slik

Kazalo tabel

Povzetek

Namen raziskovalne naloge je bil raziskati igro Zlate palice in izdelati računalniški algoritem, ki bo zmožen igro Zlate palice igrati popolno. V prvem delu naloge smo raziskali izvor igre Zlate palice in ugotovili, da je igra Zlate palice ena izmed številnih variant igre Nim. V raziskovalni nalogi smo opisali enajst variant igre Nim in razložili pravila za igranje. Postavili smo raziskovalno vprašanje, ali kljub temu, da se variante igre Nim med seboj razlikujejo, vse temeljijo na enakem matematičnem ozadju. Nato smo raziskali matematična ozadja in teorije iger enajstih variant igre Nim in ugotovili, da se posamezne variante igre Nim razlikujejo tudi v matematičnih ozadjih in teorijah iger. V drugem delu naloge smo raziskali algoritem za določanje Nim pozicij in pojasnili njegovo delovanje. Postavili smo raziskovalno vprašanje, ali je možno matematično ozadje igre Zlate palice zapisati v algoritem in ga uporabiti v računalniškem programu tako, da je računalniški program zmožen premagati vsakogar, kdor naredi že eno samo nepopolno potezo. Na to raziskovalno vprašanje smo si odgovorili, ko nam je uspelo napisati računalniški program, ki je sposoben igro Nim igrati popolno. Računalniški program smo vgradili v računalniško igro, pri kateri si igralec lahko izbere težavnost igranja, na podlagi česar program izbira med izračunanimi in naključnimi potezami. Igra je javno dostopna in jo je možno igrati preko spleta (https://novakjon.github.io).

Abstract

The aim of this research project was to investigate the game of Gold Bars and to design a computer algorithm that can play the game of Gold Bars perfectly. In the first part of the research project, we investigated the origin of the Gold Bars game and found that the Gold Bars game is one of several variants of the Nim game. In the research project, we described the eleven variants of Nim game and explained the rules for playing the game. We set a research question, are the variants of Nim game all based on the same mathematical background, although their rules differ from each other. We then investigated the mathematical backgrounds and game theories of the eleven Nim game variants and answered the research question, finding that the Nim game variants also differ in their mathematical backgrounds and game theories. In the second part of the research project, we investigated the algorithm for determining Nim positions and explained how it works. We set a research question, whether the mathematical background of the game of Gold Bars can be written into an algorithm and used in a computer program in such a way that the computer program is able to beat anyone who makes even a single imperfect move. We answered the question, by writing a computer program that is able to play the Nim game perfectly. We integrated the computer program into a computer game where the player can choose the difficulty of the game, which drives the program to choose between calculated and random moves. The game is now publicly available and can be played online (https://novakjon.github.io).

Ključne besede: Zlate palice, teorija igre, Nim pozicija, algoritem, računalniška igra

Keywords: Golden bars, game theory, Nim position, algorithm, computer game

1 Uvod

V zadnjem desetletju je industrija, ki proizvaja računalniške igre, doživela ogromen razmah, zato poznamo veliko različnih računalniških iger vseh vrst, od akcijskih, pustolovskih in strelskih iger, arkadnih iger, iger na srečo do strateških, matematičnih iger in logičnih ugank. Prav matematične igre pa so me najbolj pritegnile, zato sem se želel tudi sam preizkusiti v raziskovanju matematičnega ozadja računalniških iger in teorije iger ter v izdelavi svoje lastne matematične igre. Prišel sem na idejo, da bi izdelal igro Zlate palice, ki sem jo poznal le bežno. Tako sem se poglobil v igro Nim, iz katere izvira igra Zlate palice. Raziskal sem že znana matematična ozadja in teorije iger variant igre Nim. Že opisana matematična ozadja in teorije iger variant igre Nim, sem jih dopolnil s svojimi lastnimi teorijami iger pri variantah, kjer opisi še niso obstajali. To znanje sem to znanje uporabil za izdelavo računalniškega programa in spletne računalniške igre Zlate palice.

1.1 Cilj

Cilj te raziskovalne naloge je predstaviti matematično ozadje in teorijo igre Nim in njenih variant, vključno z igro Zlate palice, ter z uporabo matematičnega algoritma izdelati računalniški program, ki bo zmožen premagati v igri Zlate palice vsakogar, ki je ne zna igrati popolno.

1.2 Raziskovalna vprašanja

1. Ali variante igre Nim temeljijo na enakem matematičnem ozadju kljub temu, da se med seboj njihova pravila nekoliko razlikujejo?
2. Ali je matematično ozadje igre Zlate palice je možno zapisati v algoritem in ga uporabiti v računalniškem programu tako, da je računalniški program zmožen premagati vsakogar, kdor naredi že eno samo nepopolno potezo?

1.3 Pravila igre Zlate palice

Igra Zlate palice se igra v dvoje. Za igro potrebujemo poljubno število škatel, v vsako izmed njih položimo poljubno število palic. Igralec, ki je na potezi, izbere eno izmed škatel, v kateri je vsaj ena palica, in iz nje vzame kolikor palic želi. Igralec, ki je na potezi, mora vzeti vsaj eno palico iz poljubne škatle. Zmaga tisti igralec, ki vzame zadnjo palico, in so po njegovi potezi, vse škatle prazne. To je ravno nasprotno od pravil »misère« klasične igre Nim, pri kateri igralec, ki vzame zadnjo palico, izgubi. Pri igri Zlate palice začetno pozicijo določi tretja oseba, ki v igro ni vključena. V primeru, da igralca nimata dostopa do tretje osebe, morata igralca sama palice poljubno razporediti v poljubno število škatel.

Pravila igre Zlate palice lahko nekoliko variirajo. Do variacij v pravilih pa prihaja pri svobodi, ki jo imata igralca, ko sta na potezi. Pri nekaterih variantah lahko igralec na potezi vzame le eno, dve ali tri palice iz poljubne škatle. Varianta pravil, ki smo jo izbrali za izdelavo računalniške igre v okviru te raziskovalne naloge določa, da igralec ni omejen pri številu palic, ki jih vzame pri svoji potezi.

1.4 Izvor igre Zlate palice

Igra Zlate palice se je po vsej verjetnosti razvila iz igre Nim, lahko bi rekli, da je ena izmed njenih mnogih variant. Nim je igra, ki se igra na enak način kot igra Zlate palice, od igre Zlate palice razlikuje le v tem, da se za igranje uporabljajo poljubni igralni predmeti (vžigalice, palice, kamenčki ipd.), ki so razporejeni na kupe (ne nujno v škatle). Bistvena razlika je tudi ta, da ima igra Nim vedno enako začetno število in razporeditev igralnih predmetov, s katerimi igralca razpolagata. Pri originalni igri Nim so na začetku igre igralni predmeti razporejeni tako, da je na prvem kupu en predmet, na drugem trije, na tretjem pet in na četrtem sedem predmetov. Pri drugih variantah igre Nim pa so igralni predmeti na začetku igre lahko razporejeni tudi drugače, pri eni izmed variant na primer tako, da je na prvem kupu en igralni element, na drugem dva, na tretjem tri, na četrtem štiri in na petem pet igralnih predmetov.

Igra Nim ima dve različni varianti pravil za zmago. Pri prvi varianti, ki se imenuje »misère«, tisti igralec, ki vzame zadnjo palico, izgubi, pri drugi pa igralec, ki vzame zadnjo palico, zmaga. Nim se po navadi igra po pravilih bolj razširjene prve variante »misère«.

O izvoru igre Nim ne vemo prav veliko. Znana je igra Jiǎn-shízi oz. igra pobiranja kamenčkov, ki izhaja iz Kitajske (Jorgensen, 2009). Kljub temu, da ta igra zelo spominja na igro Nim, ne dokazuje njenega izvora. Najstarejša evropska omemba igre Nim je bila v šestnajstem stoletju. Svoje sedanje ime in svojo prvo teorijo igranja je igra Nim dobila leta 1901. Oba je izumil Charles L. Bouton (Bouton, 1901 – 1902). O igri Nim je kasneje nastalo še veliko teorij igranja. Prav tako tudi izvora imena nikoli niso znali v celoti razložili. Oxfordov angleški slovar ime izpelje iz nemškega glagola nimm, kar pomeni vzemi.

Zdi se, da so bila pravila igre Nim spremenjena v igro Zlate palice za namen krajšanja časa nekdanjih severno evropskih kraljev, vendar za to ni nobenih zgodovinskih dokazov.

Na Svetovni razstavi leta 1939 v New Yorku je podjetje Westinghouse prikazalo napravo, imenovano Nimatron, ki je igrala igro Nim je bila ena prvih elektronskih računalniških iger. Igra je bila na voljo za igranje obiskovalcem razstave od 11. maja 1940 do 27. oktobra 1940. V tem šestmesečnem obdobju je napravo premagalo le malo ljudi, tisti ki jim je to uspelo, pa so prejeli kovanec z napisom Nim Champ (Flesch, 1951; Kellem, 1939).

Ferranti, navdihnjen z zgodnejšim Nimatronom, je leta 1951 za Festival Britanije v Združenem kraljestvu zasnoval napravo Nimrod. To je bil zgodnji računalnik velikosti dvanajst krat devet krat pet čevljev

(= 3,7 x 2,7 x 1,5 metra), izdelan po naročilu za igranje igre Nim. Oblikoval ga je John Makepeace Bennett, zgradil pa inženir Raymond Stuart-Williams. Računalnik je omogočal obiskovalcem razstave, da so igrali igro Nim proti umetni inteligenci. Igralec je na dvignjeni plošči pritiskal gumbe, ki so ustrezali lučem na stroju, s čemer je igralec določil svojo potezo. Nato pa je Nimrod svojo potezo izračunal, pri čemer so njegove izračune vizualno predstavljale dodatne luči. Hitrost izračunov Nimroda je bilo mogoče tudi zmanjšati, s čemer so obiskovalcem festivala lahko pokazali, kaj računalnik dela. Nimrod ni bila prava videoigra, saj je uporabljala žarnice namesto zaslona z gibljivimi grafikami. Nimrod je bil namenjen predvsem predstavitvi Ferrantijevega snovanja in programerskih veščin, ne pa zabavi. Kljub temu pa so se obiskovalci festivala bolj zanimali za igranje igre kot za njeno logiko. Po prvotni razstavi v Združenem kraljestvu je bil Nimrod razstavljen tudi na Berlinskem industrijskem sejmu, preden so ga razdrli.

Igra Nim, ki se je izvajala na Nimrodu, je bila ena prvih računalniških iger, ki je imela vizualni prikaz. Dokončana je bila štiri leta po izumu najstarejše znane interaktivne elektronske igre, ki uporablja elektronski zaslon, in eno leto po računalniku Bertie the Brain, ki je igral križce in krožce na kanadskem nacionalnem sejmu leta 1950 (Baker, 2010).

Leta 1952 so inženirji Herbert Koppel, Eugene Grant in Howard Bailer iz podjetja W. L. Maxon Corporation razvili stroj za igranje igre Nim, ki je bil izdelan iz konstrukcijski kompletov za otroke, imenovanih tinkertoy. Stroj, ki je tehtal 23 kilogramov, je bil sposoben igrati proti ljudem in redno zmagovati (Cohen Harvey, 1980).

Igra Nim je bila tudi tema kolumne Martina Gardnerja o matematičnih igrah v znanstveni reviji Scientific American februarja 1958 (Morrissette, 1968).

Različica igre Nim se je pojavila tudi v francoskem filmu Last Year at Marienbad. V filmu je igra Nim uporabljena kot metafora za igro med protagonisti. Igra Nim, ki jo igrajo liki v filmu, se interpretira kot simbol moči, nadzora ali igre med spominom in resničnostjo, kar poudarja zapleteno in večplastno naravo človeških odnosov ter subjektivnost spomina in dojemanja resničnosti. Ta simbolika je del filmskega sloga in tematike, ki spodbuja razmišljanje gledalcev ter pušča odprto razlago in interpretacijo.

1.5 Motivacija za izdelavo raziskovalne naloge

Z igro Zlate palice sem se srečal pri pouku matematike, ko nam je profesor zastavil izziv, ali ga pri igri Zlate palice lahko premagamo. Igra me je tako pritegnila, da sem želel raziskati matematično ozadje in teorijo igre. Ker me je igra tako navdušila, moji soigralci pa so se igre naveličali mnogo hitreje od mene, sem želel igrati Zlate palice proti računalniku, vendar računalniške oblike igre Zlate palice nisem našel. Našel sem le računalniško igro Nim, ki pa se mi zdi mnogo manj zanimiva, saj se vedno začne z enakim številom in razporeditvijo igralnih elementov. Zato sem se izdelave računalniške igre Zlate palice lotil sam, napisal program in igro namestil na spletno stran ter tako igranje na spletu omogočil tudi drugim zainteresiranim igralcem.

2 Metode dela

Najprej sem po literaturi poiskal razpoložljive podatke o igri Zlate palice in njenem izvoru. Ker sem ugotovil, da igra Zlate palice izhaja iz igre Nim, sem poiskal tudi podatke o igri Nim in njenih variantah.

V literaturi sem poiskal razpoložljive podatke o matematičnem ozadju in teoriji igre Zlate palice ter igre Nim in njenih variant. Za tiste variante, za katere podatkov o matematičnem ozadju in teoriji igre nisem našel, sem sam razmislil, kakšno matematično ozadje in teorija igre stoji za njimi in ga predstavil v raziskovalni nalogi.

V drugem delu raziskovalne naloge sem uporabil programsko orodje Phyton za programiranje računalniške igre Zlate palice. V računalniškem programu sem uporabil matematično ozadje in matematično teorijo, ki sem jo raziskal v prvem delu raziskovalne naloge.

Spletno stran sem ustvaril s pomočjo jezika HTML, za oblikovanje pa sem uporabil jezik CSS. Igra Zlate palice se tudi na spletni strani izvaja v programskem jeziku Phyton. To sem dosegel z integracijo kode v spletno stran s pomočjo platforme Trinket (Purba et al, 2023).

3 Rezultati

3.1 Matematična ozadja in teorije iger

3.1.1 Zlate palice

Igro Zlate palice je možno igrati popolno, to pomeni, da v primeru, ko ima igralec možnost izbire ali bo na potezi prvi ali drugi, lahko s točno določenimi potezami vedno zmaga igro, kar je raziskal Charles L. Bouton v svojem delu Nim, a game with a complete mathematical theory. Za določitev popolne poteze mora igralec ugotoviti, kako narediti tako imenovano Nim pozicijo. Nim pozicija je vsaka pozicija (oz. razporeditev palic v škatlah), ki nasprotniku preprečuje zmago. Nim pozicija je na primer pozicija, ko ostaneta le še dve škatli, v vsaki pa je le še po ena palica, saj drugi igralec na svoji sledeči potezi v nobenem primeru ne more sebi zagotoviti zmage. Nove Nim pozicije ni mogoče narediti iz že obstoječe Nim pozicije. Iz vsake ne Nim pozicije pa je mogoče narediti eno ali več različnih Nim pozicij po različnih poteh.

Nim pozicija je vsaka pozicija za katero velja, da je ekskluzivna disjunkcija števil palic vseh kupov enak nič. Primer take pozicije je pozicija, ko je v prvi škatli ena palica, v drugi štiri in v tretji pet palic, ker je $1 \oplus 4 \oplus 5 = 0$.

Nim pozicije je najlažje izračunati z uporabo dvojiškega sistema. Pri tem postopku mora igralec na potezi zapisati vsa števila palic v posameznih škatlah kot števila v dvojiškem sistemu in sicer eno pod drugo tako, da so vsa dvojiška števila poravnana desno, s čemer tvori tabelo. Nato mora igralec prešteti število enk v vsakem stolpcu tabele. Igralec mora določiti, ali je število enk v posameznem stolpcu sodo ali liho. Nim pozicijo je vedno mogoče doseči z odvzemanjem palic iz škatle z največjim številom le-teh, zato igralec na potezi izbere škatlo z največ palicami kot tisto škatlo, iz katere bo odvzemal palice. Da igralec ugotovi, koliko palic mora za popolno potezo odvzeti iz škatle z največjim številom palic, mora v novo vrstico tabele napisati novo število tako, da številke prepiše iz vrstice, v kateri je zapisano število palic v škatli z največ palicami, pri čemer številko (enko oz. nulo) pusti nespremenjeno, če je v tem stolpcu sodo število enk, oziroma jo spremeni v obratno vrednost (enko v nulo in nulo v enko), če je v tem stolpcu liho število enk. V novi vrstici tabele tako dobi dvojiški zapis števila palic, ki morajo ostati v škatli, v kateri je trenutno največje število palic, da bo igralna pozicija, Nim pozicija. Zato mora igralec, če želi tvoriti Nim pozicijo, iz škatle z največjim številom palic odvzeti toliko palic, kolikor je razlika med obstoječim številom palic v škatli z največ palicami in novo nastalim številom v tabeli.

Z drugimi besedami za popolno potezo mora igralec najprej izračunati ekskluzivno disjunkcijo vseh škatel in nato to vsoto odstraniti iz škatle z največ palicami. S tem igralec zgotovi, da bodo po njegovi potezi vsi stolpci vsebovali sodo število enk. Vsak stolpec predstavlja eno izmed potenc števila dva (1, 2, 4, 8 itd.), enka v stolpcu pa pomeni, da število palic v škatli vsebuje to potenco, torej če seštejemo vse vrednosti potenc, pri katerih je v posamezni vrstici napisana ena, dobimo desetiško število, ki predstavlja število palic v škatli. V primeru, ko je število vseh enk v vseh stolpcih sodo, je število vseh posamičnih potenc števila dva, ki jih moramo sešteti, da dobimo število palic v škatli, tudi sodo. Za popolno potezo je tako potrebno odvzeti vsoto tistih potenc števila dva iz določene škatle, ki jih je v vseh škatlah skupaj liho. S tako potezo si igralec zagotovi, da bo lahko ne glede na potezo nasprotnika, po nasprotnikovi potezi lahko ponovno spremenil količino vseh potenc v soda števila. To pa vodi do zmage, saj takšna igra privede do

končne pozicije, ko so palice razporejene le še v dve škatli, v vsaki škatli pa je število palic enako, končno število vseh potenc števila dva pa je seveda ponovno sodo.

3.1.2 Gradbeni Nim

Pri varianti igre Nim, imenovani gradbeni Nim oz. »building nim« se igralca najprej dogovorita s koliko predmeti bosta igrala in koliko kupov predmetov bosta ustvarila. Igralca nato izmenično polagata po en igralni predmet na določen kup, dokler predmetov ne zmanjka. Tako igralca ustvarita začetno pozicijo. Igro prične igralec, ki ni zadnji položil igralnega predmeta na kup.

V drugem delu, torej po ustvarjeni začetni poziciji, je pri igri gradbeni Nim optimalen način igre v enak kot pri igri Zlate palice. V drugem delu ima igra namreč enaka pravila kot igra Zlate palice. Najboljši način igranja začetni del igre (to je del, ko igralca polagata igralne predmete na kupe) pa za določene primere opisuje delo Building Nim, ki so ga objavili Eric Duchêne, Matthieu Dufour, Silvia Heubach in Urban Larsson (Larsson et al., 2015). V svojem delu so avtorji zagotovili konkretne dokaze za določena števila kupov, ki ponazarjajo zmagovalne strategije, ki jih lahko uporabi igralec, ki igro začne. V svoji raziskavi pa niso odkrili splošnega argumenta za popolno igro, ki bi dokazoval, da lahko prvi igralec zmaga v vseh začetnih igrah s katerimkoli številom kupov. Raziskava poudarja, da lahko v večini primerov prvi igralec z uporabo strategije igranja visoko ali nizko učinkovito odgovarja na obrambne poskuse drugega igralca. Kljub temu pa v nekaterih primerih takšne strategije odpovedo. Takrat pa si prvi igralec lahko zmago zagotovi z odstopanjem od igre visoko ali nizko. Popolna igra za širši nabor začetnih primerov z različnim številom kupov pa ostaja neznana.

3.1.3 Igra odštevanja

Poleg izvorne igre Nim in njenih variant Zlate palice ter gradbeni Nim obstaja še mnogo drugih izpeljank. Ena izmed njih je igra odštevanja. Igra ima zelo podobna pravila kot izvorna igra Nim, edina razlika je v tem, da je določena zgornja meja števila predmetov, ki jih je mogoče odstraniti v eni potezi. Namesto odstranjevanja poljubnega števila predmetov, lahko igralec odstrani največ toliko predmetov, kot sta se soigralca v naprej dogovorila. Ta igra se običajno igra le z enim kupom.

Teorija te variante igre Nim je podobna osnovni teoriji o igri Nim, ki jo je v delu Nim, a game with a complete mathematical theory opisal Bouton C. L. Dodati je potrebno le še pravilo, ki upošteva tudi zgornjo omejitev števila predmetov, ki jih igralec pri svoji potezi lahko odstrani. Do tega pravila sem prišel s sledečim razmislekom: če lahko igralec na potezi odstrani največ r predmetov, lahko ne glede na to, kakšno potezo naredi, drugi igralec njegovo potezo »nevtralizira« tako, da pri svoji potezi dopolni njegovo število odvzetih palic do r + 1. Takšno »nevtraliziranje« potez pa drugemu igralcu koristi, če velja, da je N (število vseh palic) = b (poljubno naravno število) * (r + 1) + 1, saj to pomeni, da ko bo drugi igralec pravilno »nevtraliziral« dovolj potez prvega igralca, bo temu preostala le ena palica, ki je tudi zadnja. Vsem pozicijam, za katere velja zgornja enakost, pravimo Nim pozicije. Če je začetna pozicija Nim pozicija, lahko igro s pravilno uporabo opisane zmagovalne strategije vedno zmaga igralec, ki je na potezi drugi, medtem ko vse igre, ki se začnejo v ne Nim pozicijah, lahko zmaga prvi igralec.

3.1.4 Igra 21

Podobna igri odštevanja pa je tudi igra 21. Igra 21 se igra tako kot igra Nim »misère« s poljubnim številom igralcev, ki izmenično govorijo številke, vendar le manjše od 22. Prvi igralec začne igro tako, da reče ena, vsak igralec nato zaporedoma poveča številko za 1, 2 ali 3. Igralec, ki je prisiljen reči 21 izgubi. Zmagovalna

strategija za to različico igre Nim za dva igralca je vedno izbrati večkratnik števila 4, tako je zagotovljeno, da bo drugi igralec na koncu moral reči 21, tako kot pri zmagovalni strategiji igre odštevanja. Igra 21 se lahko igra tudi z drugimi števili, na primer igralec lahko doda največ pet in izgubi tisti, ki je prisiljen reči 34.

3.1.5 Igra 100

Zelo primerljiva različica pa je tudi igra 100. Pri tej različici dva igralca začneta pri 0 in izmenično dodajata število od 1 do 10 k vsoti. Igralec, ki doseže 100, zmaga. Zmagovalna strategija je doseči številko, v kateri so števke zaporedne (npr. 01, 12, 23, 34,...), ker tako igralec lahko nevtralizira poteze nasprotnika. Ko igralec doseže 89, lahko nasprotnik izbira le številke od 90 do 99, in naslednji odgovor lahko v vsakem primeru doseže 100.

3.1.6 Krožni Nim

Pri različici krožni Nim je poljubno število igralnih predmetov postavljenih v krog, igralca pa izmenično odstranjujeta enega, dva ali tri sosednje predmete.

Pri krožnem Nimu sem ugotovil, da igralec ki začne prvi, ne more nikoli zmagati, če drugi igralec pozna zmagovalno strategijo. Pri zmagovalni strategiji mora drugi igralec pri svoji potezi razdeliti krog igralnih predmetov, iz katerega je prvi igralec na svoji prvi potezi vzel nekaj igralnih predmetov, na dva enaka dela. Ko so vsi krogi igralnih predmetov razdeljeni na dva dela, drugi igralec na svojih potezah le zrcali poteze prvega igralca, kar pomeni, da jemlje toliko igralnih predmetov iz druge polovice istega kroga, kolikor jih je na svoji potezi vzel prvi igralec. S takšno igro ponavljanja je drugemu igralcu zmaga zagotovljena, saj prvi igralec s svojimi potezami drugemu igralcu ne more preprečiti možnosti ponavljanja njegovih potez.

3.1.7 Več kupčni Nim

Zanimiv je tudi koncept več kupčne igre Nim. Pri tej različici igre Nim je poleg odstranjevanja poljubnega števila predmetov iz enega kupa, dovoljeno odstraniti enako število predmetov iz vsakega kupa.

Za to varianto igre Nim so odkrite tudi zmagovalne strategije. Do teh vodi več različnih razmislekov, tako grafičnih kot računskih. Primer prvega za dva igralna kupa je, da narišemo mrežo, katere koordinate označuje par naravnih števil (n, m) z $n \leq m$, ki opisuje velikost obeh kupov v poziciji in tudi koordinate šahovske figure kraljice na mreži. Optimalna strategija iz ne Nim pozicije je premik figure v katerokoli dostopno Nim pozicijo, pri čemer velja, da je (0,0) Nim pozicija. Velja pa tudi, da je vsaka pozicija iz katere je mogoče doseči Nim pozicijo z enim premikom, ne Nim pozicija, in da je pozicija Nim pozicija, če vsak premik iz nje vodi v ne Nim pozicijo. To je ugotovil nizozemski matematik Willem Abraham Wythoff (Wythoff, 1907), prav tako pa je napisal tudi formuli za računanje parov koordinat Nim pozicij.

3.1.8 Nim z indeksom-k

Splošnejša oblika več kupčnega Nima se imenuje Nim z indeksom-k ali "indeks-k" Nim. Varianto je poimenoval E. H. Moore, ki jo je analiziral leta 1910, v svojem delu A generalization of the game called Nim (Moore, 1910). Pri igri Nim z indeksom-k igralci namesto odstranjevanja predmetov iz samo enega

kupa odstranijo predmete iz najmanj enega, vendar največ iz k različnih kupov. Število igralnih predmetov, ki jih je mogoče odstraniti iz vsakega kupa na potezi igralca je lahko bodisi poljubno, bodisi omejeno na neko dogovorjeno količino, tako kot pri že omenjeni igri odštevanja. Ugotovitve, do katerih je prišel E. H. Moore, se nanašajo le na omejitev števila kupov, iz katerih lahko igralec na svoji potezi palice jemlje, torej tako ne upoštevajo omejitve količine števila predmetov, ki jih igralec na svoji potezi lahko vzame iz vsakega kupa. Za to varianto je zmagovalna strategija sledeča.

Naj bo r največje število predmetov, ki ga igralec na svoji potezi lahko vzame in k največje število kupov, s katerih lahko jemlje. Pri razmisleku velja tudi, da je iz že obstoječe Nim pozicije nemogoče tvoriti novo Nim pozicijo in je zato igralcu, ki začne igro v Nim poziciji ob pravilni igri nasprotnika zmaga onemogočena. Igralec, ki je na potezi, ko je pozicija ne Nim pozicija, pa lahko po sledečem logičnem postopku tvori Nim pozicijo. Najprej mora igralec izračunati ostanek pri deljenju z r + 1 za vsak kup igralnih predmetov. Razlog za to je dejstvo, da je pozicija, pri kateri je na vsakem kupu število predmetov enako ostanku pri deljenju z r + 1, enaka obstoječi poziciji, ker lahko igralec poteze nasprotnika nevtralizira toliko časa, dokler ni pozicija enaka tisti po deljenju. Nato mora vrednost vsakega ostanka kupov pretvoriti v dvojiško število in jih zapisati enega pod drugega (poravnane na desno). Potem pa je potrebno le še izračunati ostanek pri deljenju s k + 1 za vsak stolpec števk dvojiških števil, saj tako upoštevamo tudi zgornjo omejitev števila kupov, s katerih igralec pri potezi predmete lahko jemlje. Iz ostankov pri drugem deljenju pa mora sedaj igralec logično premisliti, iz katerih kupov bo vzel koliko predmetov, namreč po njegovi potezi morajo biti vsi ostanki drugega deljenja enaki 0, če želi igralec po svoji potezi doseči Nim pozicijo.

Ker sem opazil, da takšen postopek določanja popolne poteze v nekaterih pozicijah odpove, sem premislil, katere so te pozicije in kako v teh pozicijah še vseeno določimo popolno potezo. Primer igre, pri kateri izračun popolne poteze po zgornjem postopku ni pravilen, je na primer v primeru, da je za našo igro r enak tri in k enak dve, ko imamo štiri kupe igralnih predmetov in sicer na prvem, drugem in tretjem po en predmet in na četrtem kupu štiri igralne predmete. Kljub temu, da izračun pokaže, da je ta pozicija Nim pozicija, to vsekakor ne drži, saj lahko igralec na potezi vzame en igralni predmet z enega izmed prvih treh kupov in tri igralne predmete s četrtega kupa. Pozicija, ki jo igralec tvori pri taki potezi, je sigurno Nim pozicija, to pa pomeni, da pozicija, iz katere je bila tvorjena, ni mogla biti Nim pozicija.

Ugotovil sem, da v primeru, ko so v dani poziciji tudi kupi, katerih število predmetov je večkratnik števila r + 1, zgoraj navedeni izračun popolne poteze vrne le možne kandidate za popolno potezo in ne nujno popolne poteze. Te kandidate je potrebno nato preveriti s sledečim postopkom. Če si vsak kup predstavljamo kot vsoto nekega večkratnika števila r + 1 (števila p) in ostanka manjšega od r + 1, ki je večji od nič, ter če pri kupih, katerih število predmetov je enako večkratniku števila r + 1, ta večkratnik poimenujemo š, potem velja sledeče pravilo: če je najmanjša vrednost števila p izmed vrednosti števil p vseh kupov, manjša od katerega koli števila š, izračunana pozicija ni Nim pozicija. To pomeni, da so tisti kandidati, pri katerih velja, da je število p večje ali enako številu š, Nim pozicije. Če igralec na svoji potezi upošteva tudi to zakonitost, lahko z razmislekom oceni, kateri kandidati so ustrezni in tako izvede popolno potezo.

Ta zmagovalna strategija je uporabna tudi pri več kupčni igri Nim.

3.1.9 Grundyjeva igra

V Grundyjevi igri, še eni različici igre Nim, je določeno število predmetov postavljenih na en sam začetni kup. Igralca izmenično delita kup na dva (neprazna) kupa različnih velikosti. Na primer, šest predmetov lahko razdelimo na kupa pet in ena ali na kupa štiri in dve, vendar ne na kupa tri in tri. Grundyjevo igro lahko igrate kot »misère« ali običajno igro.

Teorijo igre opisuje Sprague–Grundyjev izrek, ki sta ga skupaj z njegovim dokazom neodvisno odkrila R. P. Sprague leta 1936 in P. M. Grundy leta 1939. Izrek pravi da je vsaka nepristranska igra, ki se igra z običajnimi pravili igranja ekvivalentna igri Nim z enim kupčkom in jo zato lahko predstavimo kot edinstveno naravno število, ki je velikost tega kupčka (Sprague, 1936 in Grundy, 1939). To pomeni, da lahko opišemo poljubno pozicijo katerekoli nepristranske igre le z enim naravnim številom. To naravno število imenujemo Grundyjeva vrednost ali Nim vrednost. Na ta način lahko uporabimo matematične operacije in koncepte za analizo igre in ugotovitev najboljše strategije.

Grundyjeva vrednost števila nič je enaka nič. Za izračun Grundyjevih vrednosti poljubnega naravnega števila moramo najprej izračunati točno določeno množico naravnih števil. Ta množica vsebuje vsote parov tistih Grundyjevih vrednosti, ki pripadajo parom števil, na katere se izbrano naravno število lahko razdeli (to so pari neenakih naravnih števil, ki dajejo vsoto izbranega naravnega števila). Nato pa je potrebno izračunati vrednost mex (minimum excluded value) za to množico naravnih števil. Mex je najmanjše ne negativno celo število, ki ga dana množica ne vsebuje. Primere izračunov Grundyjevih vrednosti prikazuje spodnja slika.

0	1	2	3	4	5	6	7	8	9	10	11	…
0	0	0	1	0	2	1	0	2	1	0	2	…

0: G(0) = 0

1: mex({}) = 0

2: mex({}) = 0

3: mex(G(1)+G(2)) = mex(0+0) = mex(0) = 1

4: mex(G(1)+G(3)) = mex(0+1) = mex(1) = 0

5: mex(G(1)+G(4), G(2)+G(3)) = mex(0,1) = 2

6: mex(G(1)+G(5), G(2)+G(4)) = mex(2,0) = 1

7: mex(G(1)+G(6), G(2)+G(5), G(3)+G(4)) = mex(1,2,1) = 0

8: mex(0,1,3) = 2

9: mex(2,0,2,2) = 0

10: mex(1,2,1,1) = 0

11: mex(0,1,3,0,3) = 2

3 se deli na 1,2

4 se deli na 1,3

5 se deli na 1,4 ali 2,3

6 se deli na 1,5 ali 2,4

7 se deli na 1,6 ali 2,5 ali 3,4

8 se deli na 1,7 ali 2,6 ali 3,5

9 se deli na 1,8 ali 2,7 ali 3,6 ali 4,5

10 se deli na 1,9 ali 2,8 ali 3,7 ali 4,6

11 se deli na 1,10 ali 2,9 ali 3,8 ali 4,7 ali 5,6

Slika 1: Primeri izračunov Grundyjevih vrednosti.

Za igre, katerih pozicije lahko indeksiramo z naravnimi števili, pri računanju popolnih potez uporabljamo zaporedje Grundyjevih ali Nim vrednost. To je zaporedje, ki je prikazano tudi v spodnji tabeli, lahko dobimo, ko po vrsti zapišemo Grundyjeve vrednosti za naravna števila.

Tabela 1: Zaporedje Grundyjevih vrednosti za naravna števila.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	0	0	1	0	2	1	0	2	1	0	2	1	3	2	1	3	2	4	3	0	4	3	0	4

Zaporedje Nim vrednosti je zapisano v spletni bazi podatkov celoštevilskih zaporedij The On-Line Encyclopedia of Integer Sequences pod številko OEIS: A002188 (https://oeis.org/A002188). Kljub izračunanju zaporedja Grundyjevih števil vse do 2^{35} pa še vedno ni znano ali to kdaj začne slediti kakšnemu vzorcu.

Idealna poteza Grundyjeve igre je tista, po kateri je ekskluzivna disjunkcija oz. vsota po modulu 2 vseh Grundyjevih vrednosti kupov enak nič, torej mora igralec, ki želi na svoji potezi tvoriti Nim pozicijo, premisliti, koliko predmetov mora vzeti iz katerega kupa, da bo po njegovi potezi tak seštevek enak nič.

3.1.10 Pohlepni Nim

Tudi pohlepni Nim se lahko igra na oba načina, kot »misère« ali običajno igro. Pohlepni Nim je varianta, pri kateri so igralci omejeni na izbiro predmetov iz največjega kupa. Za to varianto je raziskana teorija igranja in iznajden sistem, po katerem si igralec lahko zagotovi zmago. Ta sta opisana v delu, ki sta ga ustvarila M. H. Albert in R. J. Nowakowski, imenovanem Nim restrictions (Albert in Nowakowski, 2004). Sistem igranja je sledeč. Naj bo število predmetov na kupu z največjim številom predmetov m in drugo največje število predmetov n. Naj bo pm število kupov z m predmeti in pn število kupov z n predmeti.

Igralec, ki je na potezi, mora zagotoviti, da je po njegovi potezi pm sodo število, saj so tiste pozicije v katerih je Nim oz. zmagovalne pozicije. To je očitno in logično. V primeru je pozicija na začetku igralčeve poteze Nim pozicija, igralec ne more odigrati popolne poteze, ki bi mu zagotavljala zmago. Iz ne Nim pozicij, pa ta tvori Nim pozicije na več načinov. V primeru, da je pm večje od 2, lahko vse predmete iz tega kupa odstranimo in tako pm zmanjšamo za 1. Če je pm = 1, torej je največji kup tudi edinstven, pa mora igralec, v primeru, da je pn liho število, zmanjšati velikost največjega kupa na n, oz. v primeru ko je je pn sodo število, največji kup popolnoma odstraniti. Takšna igra zmago zagotavlja igralcu, ki igre ne začne v Nim poziciji.

3.1.11 Bombon

Nekatere variante igre Nim pa so prilagojene tako, da so za bolj zabavne za mlajše. Varianta igre Nim imenovana bombon oz. »candy« Nim je različica igre normalnega nima, pri kateri igralci poskušajo doseči dva cilja hkrati. To sta primarni cilj, ki je vzeti zadnji igralni predmet (v tem primeru sladkarijo) in sekundarni cilj, ki je do konca igre zbrati največje možno število sladkarij. Igra je natančno opisana in raziskana v delu P play in candy Nim, ki so ga napisali Nitya Mani, Rajiv Nelakanti, Simon Rubinstein-Salzedo, Alex (Mani et al., 2018).

Obstajajo pa tudi druge izpeljanke igre Nim na primer višje-dimenzionalni Nim, besedni Nim in grafični Nim.

3.2 Računalniška igra Zlate palice

Logični postopek, s pomočjo katerega lahko igralec igra popolno igro Zlate palice, sem s pomočjo programskega jezika Python pretvoril v algoritem, ki je zmožen za vsako ne Nim pozicijo poiskati potezo, s katero lahko tvori Nim pozicijo.

Nato pa sem ta algoritem uporabil pri izdelavi igre Zlate palice, kot nasprotnika igralcu, ki igra popolno, torej sestavlja Nim pozicije, ko je to le mogoče. V primerih, ko igralec igra popolno, pa algoritem prepozna, da je dana pozicija že Nim in odigra naključno potezo, ki sem jo sprogramiral s pomočjo Pythonove funkcije random.randit(), ki vrne naključno število izmed števil, ki jih zajema območje definirano v pogojih funkcije.

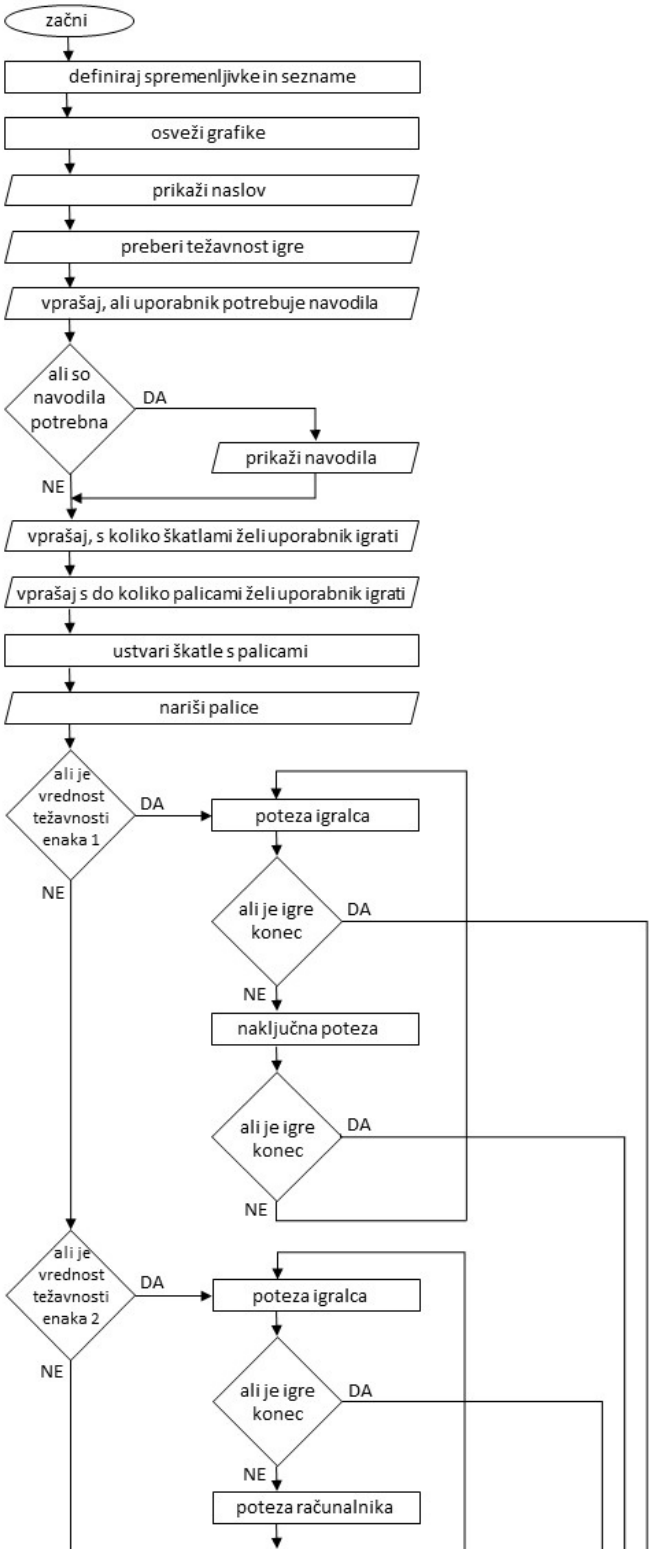
Igri sem dodal še dve lažji stopnji težavnosti, saj pri največji težavnosti računalnik igralca premaga že, če igralec le enkrat odigra nepopolno potezo. Najlažja stopnja je primerna le za tiste, ki o igri še ne vedo ničesar, saj pri tej nastavitvi računalniški program izvede vsako potezo naključno. Vmesna stopnjo, tj. drugi nivo težavnosti, je primerna za srednje vešče igralce, saj pri tej težavnostni nastavitvi računalniški program odigra svojo prvo potezo in vsako naslednjo liho potezo popolno, če je to mogoče, in vsako sodo potezo naključno. Prvotna igra, ki uporabniku olajšav ne omogoča, pa je tako postala tretji in tudi najtežji težavnostni nivo.

Računalniško igro sem opremil tudi z imenom, ki se prikaže vsakič ob začetku igre, in z razlago pravil igre, ki si jih igralec lahko prebere v primeru, da igre ne pozna. Igra igralca tudi vodi pri vsaki potezi, tako da mu jasno razloži, kako opraviti želeno potezo. Po vnesenem zahtevku za izvajanje želene poteze pa program prečrta palice, ki jih je uporabnik želel odvzeti, kot da bi igro igral na papir. Ko se igra konča, program igralcu tudi sporoči, ali je zmagal ali je izgubil. Za grafični prikaz napisov, animacij risanja zlatih palic in animacij prečrtavanja odvzetih palic iz škatel sem uporabil funkcije iz knjižnice Turtle graphics.

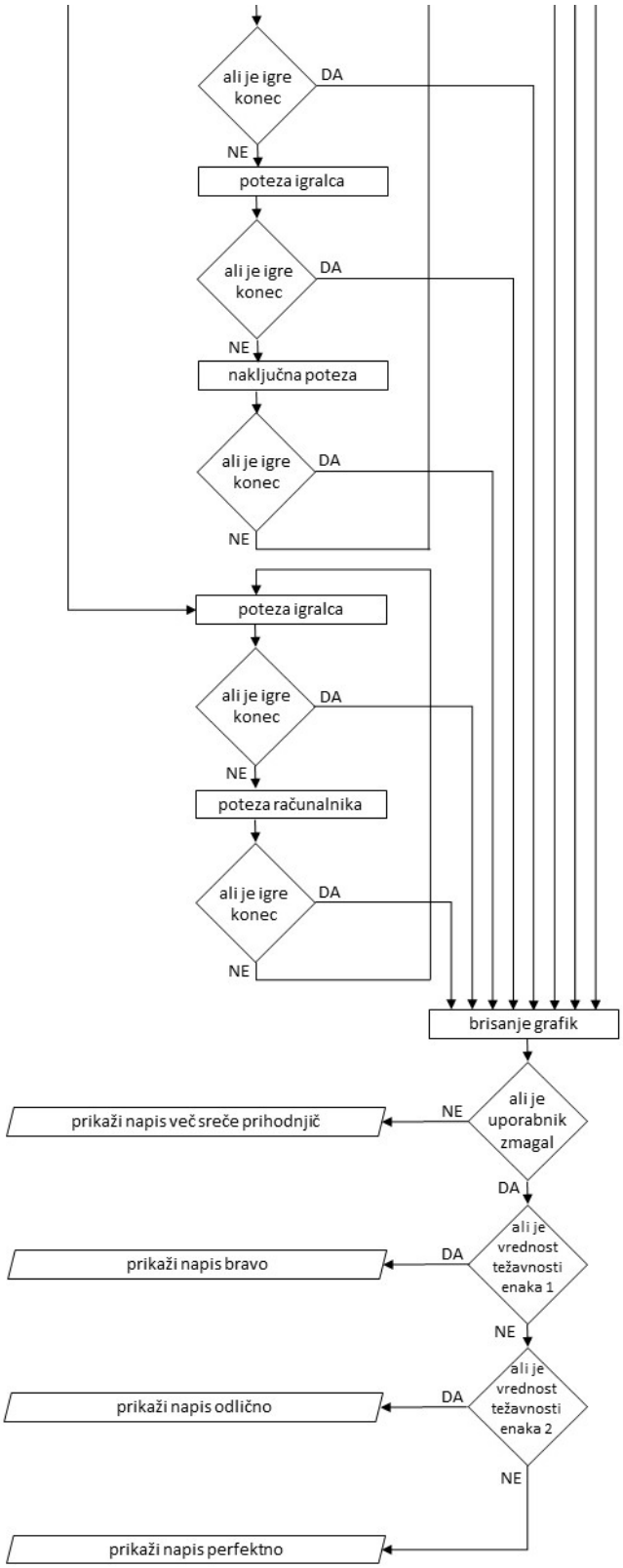
Da bi se v igri Zlatih palic lahko preizkusil kdorkoli, smo izdelali tudi spletno stran, ki vsebuje kratko predstavitev igre, opis zmagovalne strategije igranja in seveda možnost igranja igre Zlate palice. Spletna stran je tudi javno dostopna na povezavi (https://novakjon.github.io/index.html).

3.2.1 Prikaz delovanja programa

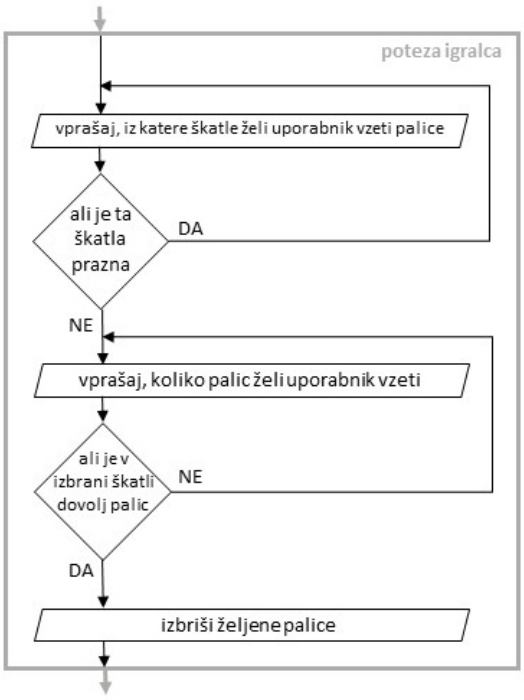
Za lažje razumevanje delovanja računalniške igre sem izdelal diagram poteka izvajanja programa, ki s pomočjo logičnih simbolov prikazuje koncepte, ki so v programu pretvorjeni v kodo. Diagram prikazuje pomembnejše dele programa, ki so v nadaljevanju tudi natančneje opisani.



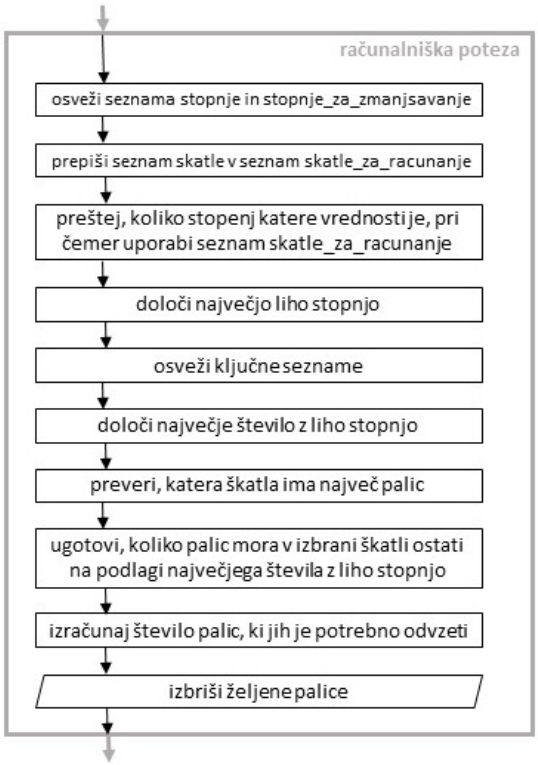
Slika 2: Diagram poteka delovanja programa.



Slika 2 (nadaljevanje): Diagram poteka delovanja programa.



Slika 3: Diagram poteka delovanja poteze igralca.



Slika 4: Diagram poteka delovanja poteze računalnika.

3.2.2 Podroben opis delovanja programa

Ko program poženemo, ta najprej definira vse potrebne spremenljivke in sezname in osveži oz. izbriše vse grafične prikaze ter je tako pripravljen za nadaljnje izvajanje. Nato program prikaže naslov, malo počaka in vpraša uporabnika ali ta potrebuje navodila za igro, ter jih prikaže, če to uporabnik želi. V nadaljevanju program uporabniku omogoči izbiro števila škatel, s kolikimi se bo igra izvajala, nato pa še izbiro največjega možnega števila palic, ki se lahko pojavi v posamičnih škatlah.

Število škatel je spodaj omejeno s tri, saj je igranje igre Zlatih palic z manj kot tremi škatlami nezanimivo, zgoraj pa je trenutno omejeno na devet škatel, ker je prikazovanje večjega števila škatel v omejenem prostoru za igro na spletni strani ne estetsko. Omejeno pa je tudi število palic in sicer spodnja meja je šest palic, ker program naključno izbere število palic za vsako škatlo, ki je pri izbiri igranja s šestimi palicami, med ena in šest in bi bilo tako igranje z manjšim številom palic ne zanimivo. Zgornja meja števila palic na posamezno škatlo pa je 31 iz podobnih razlogov, kot pri zgornji meji števila škatel.

Po uporabnikovem vnosu želenih količin, program vnesene podatke shrani in primerno obdela ter izriše palice v obliki rumenih črtic, ki so razporejene v več vrstic, ki predstavljajo posamezne škatle.

Tu se igra za uporabnika prične, uporabnik pa je tudi vedno prvi na potezi. Na svoji potezi ga program vpraša iz katere škatle želi palice jemati in koliko palic želi na svoji potezi odstraniti. Po uporabnikovem vnosu program prečrta želene palice z majhnimi poševnimi črticami, tako da igra izgleda, kot bi se igrala na papir. Sledi poteza računalniškega programa, ki pa se pri različnih težavnostnih nastavitvah razlikuje. Če je izbrana najlažja težavnostna stopnje (to je prva stopnja), program odigra naključno dovoljeno potezo, v primeru da pa je težavnost nastavljena na najtežji nivo (to je tretja težavnostna stopnja), pa program izračuna, katera poteza je v dani poziciji popolna, če je to le mogoče, torej če dana pozicija ni Nim pozicija. V primeru, da pa je dana pozicija že Nim pozicija, pa program izvede naključno potezo.

Po vsaki odigrani potezi program preveri, ali so že vse škatle prazne in bi se tako igra morala končati in preneha z zaporednim izvajanjem uporabnikovih in programsko izvedenih potez, ko je ta pogoj izpolnjen. Takrat se grafični prikazi palic izbrišejo in prikaže se končni zaslon, ki se razlikuje v primerih ko uporabnik izgubi oz. zmaga, pa tudi glede na nastavitev igre na kateri je igro uporabnik igral.

3.2.3 Algoritem izvajanja računalniške poteze

Ko pri igri pride na vrsto, tako poimenovana Pythonova funkcija oz. podprogram, *racunalniska_poteza*, se najprej izvede funkcija za brisanje določenih seznamov imenovana *refresh_stopnje_stopnje_za_zmanjsevanje*. Nato se izvede še ena funkcija, ki je zadolžena le za pripravo vseh potrebnih seznamov, za pravilno izvajanje računalniške poteze, to je *polnjenje_skatel_za_racunanje*. Ta funkcija prepiše seznam *skatle* v seznam *skatle_za_racunanje*. Potem se izvede funkcija imenovana *generacija_stopenjskih_spremenljivk*, ki prešteje koliko je stopenj katere vrednosti (to so potence števila dva oz. enice v stolpcih podpisanih dvojiških vrednosti škatel), pri tem pa uporabi seznam *skatle_za_racunanje*. Nato se v funkciji *racunalniska_poteza* izvede logični blok, ki določi največjo liho stopnjo in funkciji *dolocanje_najvecjega_stevila_z_liho_stopnjo* ter *dolocanje_skatle_za_zmanjsevanje*, ki izračunata katera škatle izmed tistih, ki vsebujejo največjo liho stopnjo je največja in tako določita iz katere škatle bo program palice odvzemal. Po določitvi želene škatle, pa program določi še koliko palic je potrebno odvzeti in število prevede iz posameznih stopenj dvojiškega sistema v desetiško število. To stori na podlagi vrednosti zapisanih v seznamih *skatle_z_liho_stopnjo*, *vrednosti_stopenj*, *stopnje_skatle_za_zmanjsevanje*, *stopnje* in *skatle* ter spremenljivke *skatla_za_zmanjsevanje*. Nakoncu pa program tudi prečrta palice, ki jih želi odstraniti.

V opis funkcije in logični bloki, ki niso namenjeni direktno računski obdelavi podatkov ali sprejemanju in vračanju podatkov uporabniku, temveč so namenjeni le brisanju, osveževanju, preverjanju razsežnosti in tipov določenih vsebin ipd. in so tako tudi pomembni za pravilno delovanje algoritma in programa na splošno, niso vključene.

Pred kodiranjem, sem zapisal algoritem s pomočjo diagrama poteka. Kodiranja sem se lotil v programskem jeziku Python. Kodo bi se dalo še bolj optimizirati. Program deluje dovolj hitro, optimizaciji v večji meri se nisem posebej posvečal, saj to ni bil namen tega projekta. Po končanem kodiranju sem poglobil svoje znanje programskega jezika Python. Program, ki ga uporablja igra Zlatih palic bi lahko deloval veliko bolj učinkovito z uporabo določenih že narejenih funkcij knjižnic in nekaterih manjših »trikov«.

3.2.4 Programska koda igre Zlate palice

Koda programa, je dostopna na povezavi (https://github.com/NovakJon/NovakJon.github.io/blob/release/zp%20koncna%20koda.py), ki se izvaja ob igranju Zlatih palic na spletni strani, pa je sledeča:

```
import random
import turtle
import time

t = turtle.Turtle()
Mpalice = 0
Ppalice = 0
a = 0
a_pomoc = 0
stopnje = []
vrednosti_stopenj = []
aREV = (len(stopnje)) + 1
b = 0
c = 0
ciklicni_x = 0
ciklicni_x_2 = 0
ciklicni_x_3 = 0
ciklicni_x_4 = 0
ciklicni_x_5 = 0
ciklicni_x_6 = 0
ciklicni_x_7 = 0
ciklicni_x_8 = 0
ciklicni_x_9 = 0
ciklicni_x_10 = 0
konec = 0
h = 0
n = 0
najvecja_liha_stopnja = 0
nakljucno_stevilo_palic = 0
nenicelna_dolzina_stopenj = 0
pomoc_2 = 0
pomoc_3 = 0
```

```
random_poteza_palce = 0
random_poteza_skatle = 0
signal = 0
signal_2 = 0
signal_3 = 0
signal_4 = 0
signal_5 = 0
signal_6 = 0
skatla_za_zmanjsevanje = 0
stevilo_palic_za_brisanje = 0
stpalic = 0
stskatel = 0
tezavnost = 0
x = 0
xREV = (len(vrednosti_stopenj)) + 1
y = 0
zmagovalec = 0
prazne_skatle = []
skatle = []
skatle_z_liho_stopnjo = []
skatle_za_racunanje = []
stopnje_skatle_za_zmanjsevanje = []
screen = turtle.Screen()
screen.setup(900, 525)


def check_input(r):
    '''Preveri ali je vnos stevilo.'''
    if r.isdigit() != True:
        while r.isdigit() != True:
            r=(input("Mislim, da si se zatipkal, poizkusi ponovno."))
    return(int(r))


def nastavitve_igre_in_start_2():
    '''Uporabnika vprasa po zelenem stevilu skatel in palic in vnos tudi
preveri.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    stskatel = input("S koliko skatlami zelis igrati?")
    stskatel = check_input(stskatel)
    while not (3 <= stskatel <= 9):
        print("Izberes lahko od 3 do 9 skatel.")
        stskatel = input("S koliko skatlami zelis igrati?")
```

```
        tskatel = check_input(stskatel)
    stpalic = input("Do koliko palic zelis na skatlo?")
    stpalic = check_input(stpalic)
    while not (6 <= stpalic <= 31):
        print("Izberes lahko od 6 do 31 palic na skatlo.")
        stpalic = input("Do koliko palic zelis na skatlo?")
        stpalic = check_input(stpalic)
    Ppalice = stpalic
    Mpalice = 1
    t.goto(400, -400)


def generacija_stopenjskih_spremenljivk():
    '''Presteje stevilo stopenj oz. presteje stevilo vsake dvojiske potence v
vseh skatlah.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    x = 0
    skatle_z_liho_stopnjo=[]
    for i in range(len(skatle)):
        x += 1
        xREV = (len(vrednosti_stopenj)) + 1
        while not skatle_za_racunanje[(x) - 1] == 0:
            xREV -= 1
            if skatle_za_racunanje[(x) - 1] - vrednosti_stopenj[(xREV) - 1] >= 0:
                stopnje[(xREV) - 1] += 1
                skatle_za_racunanje[(x) - 1] = (skatle_za_racunanje[(x) - 1] -
vrednosti_stopenj[(xREV) - 1])
                if signal == 1:
                    if xREV == najvecja_liha_stopnja:
                        skatle_z_liho_stopnjo.append(skatle[(x) - 1])


def poteza_igralca():
    '''Uporabnika vprasa iz katere skatle zeli jemati palice in koliko palic zeli
vzeti.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
```

```
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    barva = 0
    skatla_za_zmanjsevanje = input("Iz katere skatle zelis vzeti palice? (napisi
le stevilko od zgoraj navzdol)")
    skatla_za_zmanjsevanje = check_input(skatla_za_zmanjsevanje)
    if (len(skatle)) < skatla_za_zmanjsevanje:
        while not (len(skatle)) >= skatla_za_zmanjsevanje:
            skatla_za_zmanjsevanje = input("Mislim, da si se zatipkal, poizkusi
ponovno.")
            skatla_za_zmanjsevanje = check_input(skatla_za_zmanjsevanje)
    if skatle[(skatla_za_zmanjsevanje) - 1] == 0:
        while not skatle[(skatla_za_zmanjsevanje) - 1] > 0:
            skatla_za_zmanjsevanje = input("Ta skatla je ze prazna izberi
drugo.")
            skatla_za_zmanjsevanje = check_input(skatla_za_zmanjsevanje)
            if (len(skatle)) < skatla_za_zmanjsevanje:
                while not (len(skatle)) >= skatla_za_zmanjsevanje:
                    skatla_za_zmanjsevanje = input("Mislim, da si se zatipkal,
poizkusi ponovno.")
                    skatla_za_zmanjsevanje = check_input(skatla_za_zmanjsevanje)

    stevilo_palic_za_brisanje = input("Koliko palic zelis vzeti?")
    stevilo_palic_za_brisanje = check_input(stevilo_palic_za_brisanje)
    if stevilo_palic_za_brisanje > skatle[(skatla_za_zmanjsevanje) - 1]:
        while not stevilo_palic_za_brisanje <= skatle[(skatla_za_zmanjsevanje) -
1] and stevilo_palic_za_brisanje != 0:
            stevilo_palic_za_brisanje = input("V tej skatli ni toliko palic.")
            stevilo_palic_za_brisanje = check_input(stevilo_palic_za_brisanje)

    objavi_brisanje()
    zmagovalec = 1


def vse_skatle_prazne():
    '''Preveri ali so vse skatle ze prazne in s tem ali se mora igra koncati.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    signal_5 = 0
    ciklicni_x_9 = 0
    for i in range(len(skatle)):
        ciklicni_x_9 += 1
        if not skatle[(ciklicni_x_9) - 1] == 0:
            signal_5 = 1
```

```
    if signal_5 == 0:
        konec = 1


def random_poteza():
    '''Nakljucno izbere skatlo v kateri izbere nakljucno stevilo palic.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    odstranitev_nic_skatel()
    print(prazne_skatle)
    barva = 1
    signal_2 = 0
    random_poteza_skatle = random.randint(1, (len(skatle)))
    if (len(prazne_skatle)) > 0:
        while not signal_2 == 1:
            random_poteza_skatle = random.randint(1, (len(skatle)))
            signal_4 = 0
            ciklicni_x_7 = 0
            for i in range(len(prazne_skatle)):
                ciklicni_x_7 += 1
                if random_poteza_skatle == prazne_skatle[(ciklicni_x_7) - 1]:
                    signal_4 = 1
            if signal_4 == 0:
                signal_2 = 1
    if skatle[(random_poteza_skatle) - 1] == 1:
        random_poteza_palce = 1
    else:
        random_poteza_palce = random.randint(1, skatle[(random_poteza_skatle) -
1])
    pomoc_3 = 0
    signal_6 = 0
    ciklicni_x_10 = 0
    for i in range(len(skatle)):
        ciklicni_x_10 += 1
        if skatle[(ciklicni_x_10) - 1] == 0:
            signal_6 += 1
        else:
            pomoc_3 = skatle[(ciklicni_x_10) - 1]
    if (len(skatle)) - signal_6 == 1:
        random_poteza_palce = pomoc_3
    skatla_za_zmanjsevanje = random_poteza_skatle
    stevilo_palic_za_brisanje = random_poteza_palce
    objavi_brisanje()
    zmagovalec = 0
```

```
def odstranitev_nic_skatel():
    '''Ugotov katere skatle so prazne.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    ciklicni_x_6 = 0
    for i in range(len(skatle)):
        ciklicni_x_6 += 1
        if skatle[(ciklicni_x_6) - 1] == 0:
            ciklicni_x_8 = 0
            signal_3 = 0
            for i in range(len(prazne_skatle)):
                ciklicni_x_8 += 1
                if ciklicni_x_6 == prazne_skatle[(ciklicni_x_8) - 1]:
                    signal_3 = 1
            if signal_3 == 0:
                prazne_skatle.append(ciklicni_x_6)
            if (len(prazne_skatle)) == 0:
                prazne_skatle.append(ciklicni_x_6)


def racunalniska_poteza():
    '''Izracuna popolno potezo, ce je to mogoce.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    barva = 1
    signal = 0
    refresh_stopnje_stopnje_za_zmanjsevanje()
    polnjenje_skatel_za_racunanje()
    generacija_stopenjskih_spremenljivk()
    h = 0
    najvecja_liha_stopnja = 0
    ciklicni_x = (len(stopnje))
    aREV = (len(stopnje)) + 1
    while not ciklicni_x == 0:
```

```
        ciklicni_x -= 1
        aREV -= 1
        if stopnje[(aREV) - 1] % 2 > 0:
            najvecja_liha_stopnja = aREV
            ciklicni_x = 0
            h = 1
    if h == 0:
        random_poteza()
    else:
        signal = 1
        refresh_stopnje_stopnje_za_zmanjsevanje()
        polnjenje_skatel_za_racunanje()
        generacija_stopenjskih_spremenljivk()
        dolocanje_najvecjega_stevila_z_liho_stopnjo()
        dolocanje_skatle_za_zmanjsevanje()
        izracun_skatel_z_liho_stopnjo()
        izracunanje_stevila_ki_ga_zmanjsa_racunalnik()
        objavi_brisanje()
    zmagovalec = 0


def refresh_stopnje_stopnje_za_zmanjsevanje():
    '''Osvezi seznama stopnje in stopnje_za _zmanjsevanje.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    ciklicni_x_4 = (len(stopnje))
    for i in range(len(stopnje)):
        ciklicni_x_4 -= 1
        if not stopnje[(ciklicni_x_4) - 1] == 0:
            stopnje_skatle_za_zmanjsevanje[(ciklicni_x_4) - 1] = 0
            stopnje[(ciklicni_x_4) - 1] = 0


def polnjenje_skatel_za_racunanje():
    '''Prepise seznam skatle v seznam skatle_za_racunanje.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
```

```
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    y = 0
    for i in range(len(skatle)):
        y += 1
        skatle_za_racunanje[(y) - 1] = skatle[(y) - 1]


def dolocanje_najvecjega_stevila_z_liho_stopnjo():
    '''Doloci najvecji element v seznamu skatle_z_liho_stopnjo.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    while not (len(skatle_z_liho_stopnjo)) == 1:
        if skatle_z_liho_stopnjo[0] > skatle_z_liho_stopnjo[1]:
            del skatle_z_liho_stopnjo[1]
        else:
            del skatle_z_liho_stopnjo[0]


def dolocanje_skatle_za_zmanjsevanje():
    '''Preveri katera skatla ima najvec palic.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    ciklicni_x_5 = 0
    if (len(skatle)) == 1:
        skatla_za_zmanjsevanje = 1
    else:
        for i in range(len(skatle)):
            ciklicni_x_5 += 1
            if skatle[(ciklicni_x_5) - 1] == skatle_z_liho_stopnjo[0]:
                skatla_za_zmanjsevanje = ciklicni_x_5


def izracun_skatel_z_liho_stopnjo():
    '''Ugotovi koliko palic je potrebno odvzeti pri popolni potezi.'''
```

```
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    pomoc_2 = skatle_z_liho_stopnjo[0]
    xREV = (len(vrednosti_stopenj)) + 1
    while not xREV == 0:
        xREV -= 1
        if (skatle_z_liho_stopnjo[0] - vrednosti_stopenj[(xREV) - 1]) >= 0:
            stopnje_skatle_za_zmanjsevanje[(xREV) - 1] += 1
            skatle_z_liho_stopnjo[0] = skatle_z_liho_stopnjo[0] -
vrednosti_stopenj[(xREV) - 1]


def izracunanje_stevila_ki_ga_zmanjsa_racunalnik():
    '''Doloci koliko palic bo odvzetih.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    b = 0
    if (len(skatle)) == 1:
        stevilo_palic_za_brisanje = skatle[0]
        skatla_za_zmanjsevanje = 1
    else:
        while not b == (len(stopnje_skatle_za_zmanjsevanje)):
            b += 1
            if stopnje[(b) - 1] > 0:
                if stopnje[(b) - 1] % 2 == 0:
                    if stopnje_skatle_za_zmanjsevanje[(b) - 1] == 1:
                        skatle_z_liho_stopnjo[0] = skatle_z_liho_stopnjo[0] +
vrednosti_stopenj[(b) - 1]
                else:
                    if stopnje_skatle_za_zmanjsevanje[(b) - 1] == 0:
                        skatle_z_liho_stopnjo[0] = skatle_z_liho_stopnjo[0] +
vrednosti_stopenj[(b) - 1]
            stevilo_palic_za_brisanje = pomoc_2 - skatle_z_liho_stopnjo[0]


def tezavnost_igre():
```

```
    '''Vprasa uporabnika na kateri tezavnostni nastavitvi zeli igrati.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    screen.bgpic('backdrop5.1.png')
    screen.update()
    tezavnost = input("Klikni v to polje in izberi tezavnost igre.")
    tezavnost = check_input(tezavnost)
    while not 0 < tezavnost < 4:
        tezavnost = input("Ta tezavnost ni mogoca.")
        tezavnost = check_input(tezavnost)


def generiranje_vrednosti_stopenj():
    '''Izracuna potence stevila dva.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    a_sen_pac = 0
    for i in range(9):
        vrednosti_stopenj.append(2**a_sen_pac)
        a_sen_pac += 1
        stopnje.append(0)
        stopnje_skatle_za_zmanjsevanje.append(0)


def objavi_start():
    '''Pripravi figuro t za premikanje.'''
    t.hideturtle()
    t.penup()


def objavi_izbrisi_svincnik():
    '''Izbrise vse kar je narisala figura t.'''
    t.clear()


def zp():
```

```
    '''Narise zlato palico.'''
    t.pendown()
    t.pensize(6)
    t.color("gold")
    t.hideturtle()
    t.setheading(90)
    t.speed(2)
    t.forward(20)
    t.speed(0)
    t.penup()


def objavi_risanje_palic():
    '''Doloci mesta kjer bodo narisane zlate palice.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    n = 0
    for i in range(stskatel):
        t.goto(-350, (165 - (37 * n)))
        n += 1
        nakljucno_stevilo_palic = 0
        nakljucno_stevilo_palic = random.randint(Mpalice, Ppalice)
        for i in range(nakljucno_stevilo_palic):
            t.speed(0)
            t.penup()
            t.setheading(0)
            t.forward(20)
            t.setheading(270)
            t.forward(20)
            zp()
        skatle.append(nakljucno_stevilo_palic)
        skatle_za_racunanje.append(nakljucno_stevilo_palic)


def objavi_brisanje():
    '''Doloci mesta kjer bodo palice precrtane.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
```

```
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    if barva == 1:
        t.goto((-330 + (20 * (skatle[(skatla_za_zmanjsevanje) - 1]))),(156 - (37
* (skatla_za_zmanjsevanje - 1))))
        for i in range(stevilo_palic_za_brisanje):
            t.speed(0)
            t.setheading(180)
            t.forward(20)
            sprite1()
        skatle[(skatla_za_zmanjsevanje) - 1] = (skatle[(skatla_za_zmanjsevanje) -
1] - stevilo_palic_za_brisanje)
    else:
        druga_figura()


def druga_figura():
    '''Doloci mesta kjer bodo palice precrtane.'''
    global stopnje_skatle_z_liho_stopnjo, barva, Mpalice, Ppalice, a, a_pomoc,
stopnje, vrednosti_stopenj, aREV, b, c, ciklicni_x, ciklicni_x_2, ciklicni_x_3,
ciklicni_x_4, ciklicni_x_5, ciklicni_x_6, ciklicni_x_7ciklicni_x_9,
ciklicni_x_10, konec, n, najvecja_liha_stopnja, nakljucno_stevilo_palic,
nenicelna_dolzina_stopenj, pomoc_2, pomoc_3, random_poteza_palce,
random_poteza_skatle, signal, signal_2, signal_3, signal_4, signal_5, signal_6,
skatla_za_zmanjsevanje, stevilo_palic_za_brisanje, stpalic, stskatel, tezavnost,
x, xREV, y, zmagovalec, prazne_skatle, skatle, skatle_z_liho_stopnjo,
skatle_za_racunanje, stopnje_skatle_za_zmanjsevanje, ciklicni_x_11, posebnost_1,
posebnost_2, signal_7, lovro_1
    t.goto((-330 + (20 * (skatle[(skatla_za_zmanjsevanje) - 1]))),
           (156 - (37 * (skatla_za_zmanjsevanje - 1))))
    for i in range(stevilo_palic_za_brisanje):
        t.speed(0)
        t.setheading(180)
        t.forward(20)
        sprite2()
    skatle[(skatla_za_zmanjsevanje) - 1] = (skatle[(skatla_za_zmanjsevanje) - 1]
- stevilo_palic_za_brisanje)


def sprite1():
    '''Precrta zlato palico.'''
    t.pendown()
    t.pensize(5)
    t.hideturtle()
    t.color("blue")
    t.penup()
    t.setheading(135)
    t.forward(4)
    t.pendown()
    t.setheading(315)
    t.speed(1)
    t.forward(8)
```

```
    t.speed(0)
    t.setheading(135)
    t.forward(4)
    t.setheading(180)
    t.penup()


def sprite2():
    '''Precrta zlato palico.'''
    t.pendown()
    t.pensize(5)
    t.hideturtle()
    t.color("green")
    t.penup()
    t.setheading(45)
    t.forward(4)
    t.pendown()
    t.setheading(225)
    t.speed(1)
    t.forward(8)
    t.speed(0)
    t.setheading(45)
    t.forward(4)
    t.setheading(180)
    t.penup()


objavi_izbrisi_svincnik()
generiranje_vrednosti_stopenj()
ciklicni_x_11 = 0
posebnost_1 = 0
posebnost_2 = 0
signal_7 = 0
lovro_1 = 0

objavi_start()
t.hideturtle()
screen = turtle.Screen()
t.speed(0)
t.penup()
screen.bgpic('backdrop1.1.png')
screen.update()
time.sleep(2)
screen.bgpic('backdrop8.1.png')
screen.update()
tezavnost_igre()
screen.bgpic('backdrop4.1.png')
screen.update()
lovro_1 = str(input("Napisi navodila ali pa pritisni le enter."))
if lovro_1.lower() == "navodila":
    t.goto(-327, -80)
    screen.bgpic('backdrop9.1.png')
```

```
    screen.update()
    print("Dobro preberi navodila.")
    time.sleep(15)
screen.bgpic('backdrop10.1.png')
screen.update()
nastavitve_igre_in_start_2()
screen.bgpic('backdrop8.1.png')
objavi_risanje_palic()
generacija_stopenjskih_spremenljivk()
if tezavnost == 1:
    while not konec == 1:
        if konec == 0:
            poteza_igralca()
            time.sleep(1)
            vse_skatle_prazne()
        if konec == 0:
            random_poteza()
            time.sleep(1)
            vse_skatle_prazne()
if tezavnost == 2:
    while not konec == 1:
        if konec == 0:
            poteza_igralca()
            time.sleep(1)
            vse_skatle_prazne()
        if konec == 0:
            racunalniska_poteza()
            time.sleep(1)
            vse_skatle_prazne()
        if konec == 0:
            poteza_igralca()
            time.sleep(1)
            vse_skatle_prazne()
        time.sleep(2)
        if konec == 0:
            random_poteza()
            time.sleep(1)
            vse_skatle_prazne()
if tezavnost == 3:
    while not konec == 1:
        if konec == 0:
            poteza_igralca()
            time.sleep(1)
            vse_skatle_prazne()
        if konec == 0:
            racunalniska_poteza()
            time.sleep(1)
            vse_skatle_prazne()
objavi_izbrisi_svincnik()
objavi_start()
t.goto(400, -400)
t.clear()
```

```
if zmagovalec == 1:
    if tezavnost == 1:
        screen.bgpic('backdrop2.1.png')
        screen.update()
    if tezavnost == 2:
        screen.bgpic('backdrop7.1.png')
        screen.update()
    if tezavnost == 3:
        screen.bgpic('backdrop6.1.png')
        screen.update()
else:
    screen.bgpic('backdrop3.1.png')
    screen.update()
```

4 Zaključek

Menim, da je bil cilj raziskovalne naloge dosežen, saj sem tekom njene izdelave raziskal igro Zlate palice, njen izvor in njeno matematično teoretično ozadje. Ugotovil sem, da je igra Zlate palice ena izmed mnogih variant igre Nim, ki so nastale zaradi majhnih sprememb v pravilih originalne igre Nim. Zelo zanimivo je, da nekatere spremembe v pravilih igre ne privedejo do večjih sprememb matematičnih podlag, ki opisujejo popolno igro, medtem ko druge spremembe pravil igre privedejo do bistveno drugačnega teoretičnega ozadja. Tako sem si odgovoril na svoje prvo raziskovalno vprašanje.

Izpolnil sem tudi drugi cilj raziskovalne naloge, namreč uspešno sem izdelal računalniški program, ki je zmožen igro Zlate palice igrati popolno. Program temelji na algoritmu za določanje Nim pozicij, ki je vanj integriran. S tem sem si odgovoril na svoje drugo raziskovalno vprašanje. Računalniški program sem vgradil v računalniško igro, ki ima veliko možnih nastavitev, med katerimi lahko uporabnik izbira. Računalniška igra je uporabniku prijazna, saj ga tekom igranja vodi. Izdelal sem tudi javno dostopno spletno stran, na kateri se je mogoče preizkusiti v igri Zlate palice (https://novakjon.github.io).

5 Viri

Albert M. H., Nowakowski R. J. (2004). *NIM restrictions*. Electronic journal of combinatorial number theory 4: 1-10.

Bouton C. L. (1901 – 1902). *Annals of Mathematics*. Second Series, 3 (1/4), 35-39.

Baker C. (2010). *Nimrod, the World's First Gaming Computer*. Pridobljeno 3. 3. 2024 s https://www.wired.com/2010/06/replay/.

Cohen Harvey A. (1980). *How to Construct NIM Playing Machine.* Learn to Love Mathematics, Mathematics Association of Victoria, 336-350.

Flesch R. (1951). *The Art of Clear Thinking*. New York: Harper and Brothers Publishers. p. 3.

Grundy, P. M. (1939). Mathematics and games. Eureka, 2, 6–8.

Jorgensen A. H. (2009). *Context and driving forces in the development of the early computer game Nimbi*. IEEE Annals of the History of Computing, 31 (3), 44–53.

Kellem B. (1939). *The Nimatron*. JSTOR Daily.

Larsson U., Heubach S., Dufour M., Duchêne E. (2015). *Building Nim*. arXiv:1502.04068.

Mani N., Nalakanti R., Rubinstein-Salzedo S., Tholen A. (2018). *P Play in Candy Nim*. arXiv:1805.07019.

Moore E. H. (1910). *A Generalization of the Game Called Nim. Annals of Mathematics*, Second Series, 11 (3), 93-94.

Morrissette B. (1968). *Games and game structures in Robbe-Grillet*, Yale French Studies (41), 159–167.

Purba H. S. , Adini M. H., Wiranda N., Riduan A., Sukmawati R. A., Maulana M. (2023). *Trinket Integration in the Development of Problem- Based Learning Interactive Media on Python Programming Materials*. International Journal of Innovative Science and Research Technology, 8 (9), 1523- 1526.

Sprague, R. P. (1936). *Über mathematische Kampfspiele*. Tohoku Mathematical Journal, 41, 438–444.

Wythoff, W. A. (1907). *A modification of the game of nim*, Nieuw Archief voor Wiskunde, 7 (2), 199–202.