LETNIK 7 (2023) – ŠT. 1 ISSN 2536-3565

Univerza v Mariboru

Fakulteta za naravoslovje in matematiko

D I A N O I A

REVIJA ZA UPORABO NARAVOSLOVNO-MATEMATIČNIH ZNANOSTI

ISSN	**2536-3565**
Naslov publikacije/Title	**DIANOIA**, revija za uporabo naravoslovnih in matematičnih znanosti **DIANOIA**, journal for applications of natural and mathematical sciences
Letnik/Volume	**7**
Leto/Year	**2023 (april)**
Številka/Number	**1**
Založnik in izdajatelj/	Univerzitetna založba Univerze v Mariboru, Slomškov trg 15, 2000 Maribor, Slovenija,
Published & Issued by	http://press.um.si/, zalozba@um.si
Uredništvo/Editorial board	*odgovorni urednik/editor in chief* Mitja Slavinec *glavni urednik/executive editor* Drago Bokal *izvršna urednica/managing editor* Janja Jerebic *urednici za področje biologije/editors for biological sciences* Nina Šajna, Sonja Škornik *urednik za področje didaktike/editor for didactical sciences* Samo Repolusk *urednika za področje fizike/editors for physical sciences* Robert Repnik, Aleš Fajmut *urednika za področje matematike/editors for mathematical sciences* Igor Pesek, Janja Jerebic *urednik za področje tehnike/editor for technical sciences* Mateja Ploj Virtič *tehnična urednica/technical editor* Polona Kren
Mednarodni uredniški svet/ International advisory board	Igor Emri (Fakulteta za strojništvo Univerze v Ljubljani, član SAZU), Matej Brešar (FNM, član SAZU), Sergey Pasechnik (Državna fakulteta v Moskvi), Vlad Popa-Nita (Fakulteta za fiziko Univerze v Bukarešti), Blaž Zmazek (FNM), Samo Kralj (FNM), Franci Janžekovič (FNM), Nataša Vaupotič (FNM), Mitja Kaligarič (FNM), Boris Aberšek (FNM), Andrej Šorgo (FNM), Bojan Mohar (Simon Fraser University, Vancouver), Matjaž Perc (FNM), Ivica Aviani (Naravoslovno matematična fakulteta Split), Fahriye Altınay (Univerza v Nikoziji), Andreas M. Hinz (Univerza Ludwig-Maximilians, München)
Oblikovanje/Design	Amadeja Bratuša
Lektoriranje/Proofreading	Ljudmila Bokal
Sedež uredništva/Address	FNM UM, Koroška cesta 160, 2000 Maribor
e-mail	dianoia@um.si
internet/web	www.fnm.um.si
Tisk/Printed by	FNM UM
Leto izida/Year	2023
Datum natisa/Published	2023
Naklada/Nr. of Copies	100 izvodov

Revija izhaja dvakrat letno, predvidoma aprila in septembra.

Kazalo / Table of Contents

Od ideje do realizacije

Mitja Slavinec 5

Povezava med recipročnim učenjem s spleta (RUS) in problemskim učenjem
The connection between Internet reciprocal teaching (IRT) and problem-based learning

Dejan Zemljak, Maja Kerneža 9

Nepričakovana spojina
Unexpected compound

Sara Gorenjak, Amalija Golobič, Brina Dojer 19

Večnitne simulacije in modularni preizkusi v programu Python
Multithreaded simulations of emotional states and unit testing in Python

Maša Galun, Ema Smolič, Drago Bokal 27

Povabilo v prostore znanja
An invitation to knowledge spaces

Maša Galun, Drago Bokal 41

Uporaba binarnega protokola pri testiranju vzorcev krvi
Application of Binary protocol in testing blood samples

Lara Prijatelj 55

UVODNIK/EDITORIAL

Dianoia 7 (2023) 05-07

Od ideje do realizacije

Mitja Slavinec
Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Koroška cesta 160, 2000 Maribor, Slovenija

Spomnim se pogovora izpred osmih let z Dragom Bokalom, ko se je rodila ideja o Dianoi. Ves čas sem si želel, da bi na naši Fakulteti za naravoslovje in matematiko imeli svojo revijo. Še toliko bolj, saj sem nekaj izkušenj na tem področju že imel, po drugi strani pa sem dobro vedel, koliko dela to pomeni. Pred dvema desetletjema smo na Oddelku za fiziko že pripravili osnutek in prve članke naše fizikalne revije, ki je predvsem po zaslugi Roberta Repnika imela tudi veliko astronomskega pridiha, kar so zelo hvaležne vsebine za popularizacijo naravoslovja. Splet okoliščin je takrat nanesel, da revija ni zagledala luči sveta. Ideja in želja pa je seveda ostala in tiho tlela naprej.

Fakultetno revijo sem dojemal kot ustvarjanje okolja, v katerem bodo študenti dobili ene prvih izkušenj in se bodo lahko mojstrili v strokovnem pisanju. Seveda je za kaj takega na svetu veliko že obstoječih revij, vendar prve korake je zmeraj lažje in bolj varno delati v varnem okolju domačega doma. Ko sem videl, da se tudi nekatere druge fakultete na naši Univerzi v Mariboru ponašajo s svojimi revijami, je odločitev padla.

Zelo me veseli, da sva z Dragom že na samem začetku bila na istem imenovalcu. Še več, takoj je predlagal številne nove ideje, njegov je tudi predlog imena, ki še sedaj marsikomu dela težave pri izgovorjavi, ampak ravno zato se je mnogim zdelo primerno. Zelo sem mu hvaležen, da je prevzel naloge glavnega urednika, še bolj pa, da je uspel okrog sebe ustvariti ekipo zanesenjakov, v glavnem njegovih študentk, vključil pa je tudi družino, ki so vsi projekt vzeli za svojega in brez katerih revija zagotovo ne bi bila taka, kot je. Vprašanje, če bi sploh dočakala sedmi letnik z vsemi številkami, izdanimi ob vnaprej dogovorjenih terminih. Prva številka je izšla ob Dnevu Fakultete za naravoslovje in matematiko. Tudi vsa naslednja leta je ob Dnevu FNM izšla prva številka tistega letnika. To je moč zagotoviti le ob dobrem načrtovanju in seveda še pomembneje, ob zadostnem številu prispelih člankov.

Ekipa je znala motivirati študente, da so praviloma po zagovoru svojih diplom ali magistrskih nalog še zbrali motivacijo in vsebino predelali v obliko za objavo v Dianoi.

Na začetku je zmeraj ideja. Idej je veliko, vendar jih je le malo realiziranih, saj je na poti od ideje do realizacije treba še marsikaj narediti. Najtežje je zmeraj na začetku. Pripraviti je treba obliko revije, njeno postavitev, vpis v baze, uskladiti tiskanje, zagotoviti tehnično urejanje, lektoriranje, spletne objave, distribucijo in tudi pozicioniranje med bralci. Vse to je ob naslednji številki že pripravljeno. Tudi članke je za vsako naslednjo številko lažje pridobiti kot za prvo, ko avtorji morajo med drugim izkazovati tudi veliko mero zaupanja v projekt, ki še ne živi. Uspeh prve revije vse sodelujoče navduši in motivira za naprej. S časom vse postane lažje, na nek način tudi rutinirano, kar pa je lahko past, da revija postane sama sebi namen. To se v primeru

E-mail naslov/i: mitja.slavinec@um.si (Mitja Slavinec)

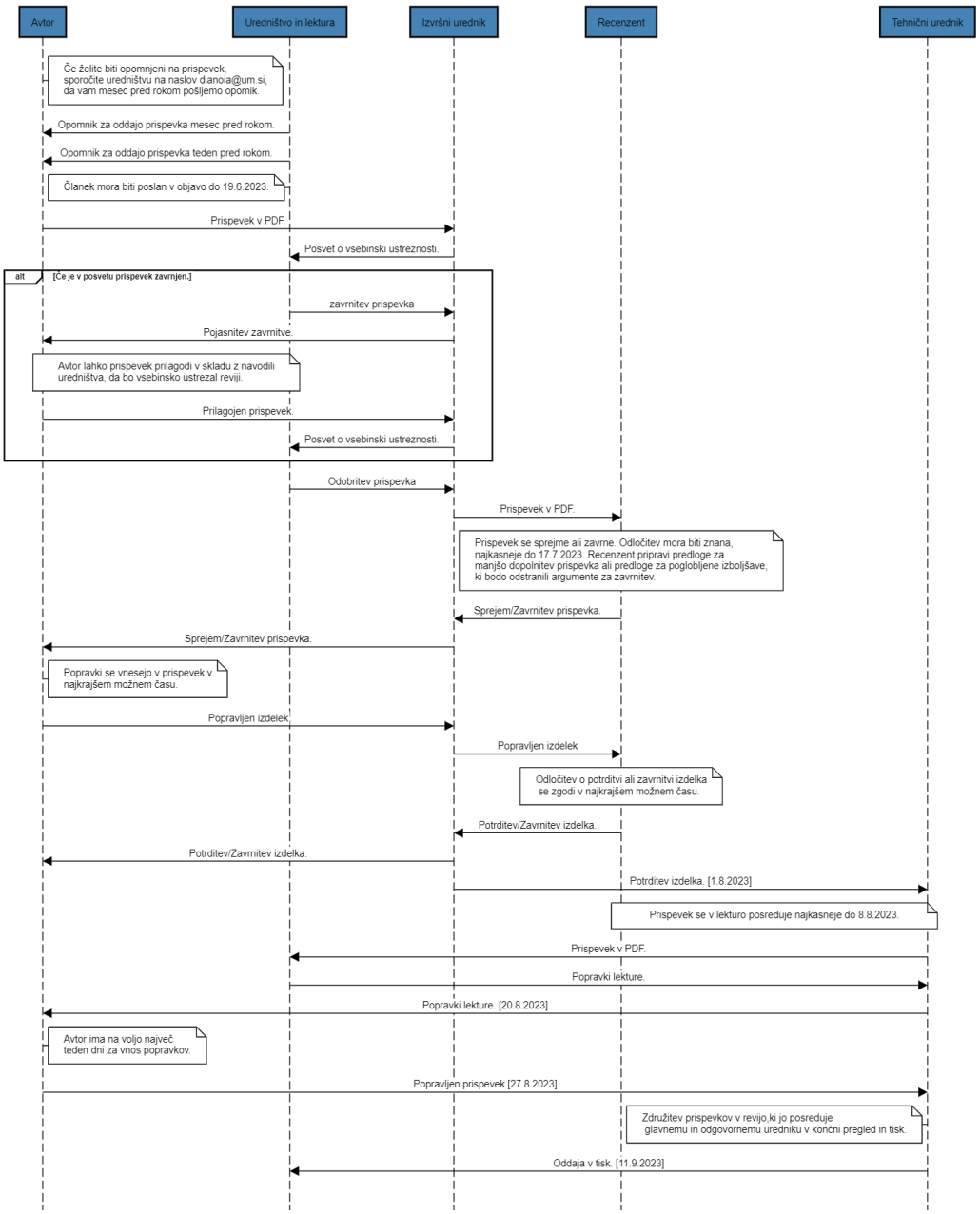
Dianoie seveda ni zgodilo in k temu je veliko pripomogla vztrajnost, ki jo v svojem članku tako lepo opisuje nekdanja tehnična urednica Monika Vogrinec [1]. Vidimo, da je materialna sredstva v bistvu še najlažje zagotoviti, ključna je zmagovita ekipa z visoko postavljenimi cilji in prepričanjem, da jih bodo dosegli.

Lepo se zahvaljujem in čestitam vsem, ki ste ustvarili vse dosedanje številke revije in hvala tudi vnaprej za vse, ki še prihajajo. Največja zahvala pa tudi motivacije so zagotovo številni pomembni in zanimivi članki, s katerimi je Dianoia obogatila zakladnico našega znanja.

Naj se ob zaključku tega uvodnika ponovno spomnim pogovora, s katerim sem uvodnik pričel. Z Dragom sva se pogovarjala, da bo to le prva revija, ki bo na nek način utrla pot in pripravila osnove za naslednjo, zastavljeno še bolj ambiciozno.

Literatura

[1] Vogrinec M. Ko vztrajnost uniči strah pred neuspehom in ne obratno, Dianoia 5, (2021) 75-76.



Slika 1: Proces izdaje naslednje številke revije Dianoia. Vir: lasten.

STROKOVNI PRISPEVKI

Dianoia 7 (2023) 09-17

Povezava med recipročnim učenjem s spleta (RUS) in problemskim učenjem

The connection between Internet reciprocal teaching (IRT) and problem-based learning

Dejan Zemljak[1], Maja Kerneža[2]

[1]*Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Koroška cesta 160, 2000 Maribor*

[2]*Univerza v Mariboru, Pedagoška fakulteta, Koroška cesta 160, 2000 Maribor*

Povzetek

Spremembe v sodobnem učenju in poučevanju so nas pripeljale do novih pogledov in pristopov v izobraževanju. K temu je pripomogla tudi pandemija COVID-19, zaradi katere se je učenje in poučevanje iz šolskega praviloma preselilo v domače okolje. Zaprtje knjižnjic je za udeležence učenja in poučevanja pomenilo, da so morali poiskati alternativne vire, ki so jim bili dostopni v času izolacije. Pri tem je pomembno vlogo prevzel svetovni splet, ki je bil eden ključnih virov informacij že prej, a se je pomen spleta še poglobil. Udeleženci učenja in poučevanja so bili tako izpostavljeni številnim virom in informacijam, med katerimi so tudi takšni, ki niso povsem resnični, so nepopolni, nepreverjeni in tudi izmišljeni. V prispevku predstavljamo metodo recipročnega učenja s spleta, ki učiteljem daje orodje za poučevanje v spletnih učnih okoljih, učencem pa znanje kritičnega vrednotenja virov in informacij s spleta. Vpeljavo predstavljamo na primeru problemskega pouka in izpostavljamo ključne vidike povezanosti metode in problemskega pouka.

Ključne besede: kritično vrednotenje informacij, izobraževanje, problemski pouk, recipročno učenje s spleta.

Abstract

Changes in modern learning and teaching have led us to new views and approaches in education. The COVID-19 pandemic also contributed to this, as a result of which learning and teaching generally moved from the school classroom to the home environment. The closure of libraries meant that learning and teaching participants had to find alternative resources that were accessible to them during the time of isolation. The World Wide Web, which was one of the key sources of information earlier, played an important role in this, but the importance of the Web has deepened. The participants in learning and teaching were thus exposed to many sources and information, including such sources and information that are not entirely true, are incomplete, unverified and even fabricated. In this paper, we present the possibility of how we can integrate the method of reciprocal learning from the web into learning and teaching, which helps students to critically evaluate resources and information from the web. We present the introduction on the example of problem-based teaching and highlight the key aspects of the connection between the method and problem-based teaching.

Key words:critical evaluation of information, education, problem-based learning, internet reciprocal teaching.

E-mail naslov: dejan.zemljak1@um.si (Dejan Zemljak), maja.kerneza1@um.si (Maja Kerneža)

1 UVOD

Poročilo The Short and Winding Road to 2030: Measuring Distance to the SDG Targets [9][1] izpostavlja 17 ciljev, ki so neposredno vezani na to, kakšno bo življenje v prihodnosti. Pri tem se cilj 4 naposredno navezuje na področje izobraževanja in poziva države, da naj zagotovijo vsem ljudem možosti vključevanja v izobraževanje, naj zagotovijo kakovostno izobraževanje in omogočajo vseživljenjsko učenje za vse. Hkrati ta publikacija poudarja, da vse države do leta 2030, ko je predvideno, da bomo dosegli zastavljene cilje na področju izobraževanja, teh ciljev ne bodo dosegle. Razlog je, da je na svetu preveč takšnih otrok, mladih in odraslih, ki nimajo osnovnih veščin, ki so potrebne, da bi postali angažirani državljani in živeli boljše življenje [9].

Svoje je k odmiku dosega ciljev in nenazadnje tudi k poslabšanju stanja dodala kriza, povezana s pandemijo COVID-19, ki je močno prizadela izobraževanje. Šole so bile nekaj časa zaprte, izobraževanje je bilo oteženo in podvrženo novim večini ljudem, neznanim situacijam. Izobraževanje je potekalo na daljavo, pri čemer je pomembno vlogo igrala tehnologija, učenci so se v času izobraževanja na daljavo morali zanašati tudi na lastna sredstva. Ne samo v obliki opreme, temveč tudi v obliki virov informacij, saj so bile tudi knjižnice nedostopne [10]. In tako je eden najpogostejših virov informacij postal svetovni splet, ki predstavlja tako rekoč sodobno knjižnico informacij. A največja težava pri iskanju informacij preko spleta je, da smo lahko hitro podvrženi neresničnim, nepopolnim in neverodostojnim informacijam, ki jih težko prepoznamo kot neustrezne že odrasli, kaj šele otroci. Zato se nam zdi pomembno, da mlade opolnomočimo kritičnega mišljenja in razmišljanja, ki jim bo omogočilo, da pred uporabo kritično presodijo, če so informacije, pridobljene s spleta, relevantne in kot take tudi resnične.

Iskanje informacij po spletu je danes postalo del vsakdanjega izobraževanja v šoli, tudi kot del sodobnih učnih oblik in metod dela. Zato se nam zdi pomembno, da pogledamo, kako lahko učitelji prispevajo k opolnomočenju mladih za kritično mišljenje ob iskanju informacij po spletu in hkrati omogočajo sodobno naravnan pouk. V nadaljevanju zato predstavljamo primer sodobnega izobraževalnega modela, ki učencem omogoči, da najprej spoznajo pomen kritičnega mišljenja ob iskanju informacij po spletu, ob tem pa so deležni sodobnih učnih oblik dela, ki hkrati krepijo kompetence 21. stoletja.

2 SODOBNI POUK PODPRT Z IKT

Danes v učni proces vse bolj vključujemo sodobne tehnologije, ki so večinoma podprte z informacijsko-komunikacijsko tehnologijo (IKT). Hkrati vse bolj vključujemo različne metode in strategije, kot sta poizvedovanje in problemsko učenje v kombinaciji s sodelovalnim poučevanjem/učenjem in možganskimi tehnikami, ki temeljijo na informacijskih in komunikacijskih tehnologijah. S takšnim poučevanjem dobimo tudi možnost opazovanja morebitnih napak ter lastnega razmišljanja in zavedanja učencev [1]. Seveda je potreben čas za razmislek o tem, kakšne oblike učenja uporabiti pri pouku, da bomo čim lažje krepili socialne veščine in s tem opolnomočili mlade za bolj trajnostno ravnanje. Kot eno od možnosti bomo opisali primer problemskega učenja. Predstavili bomo tudi, kako lahko pri tem uporabimo metodo recipročnega učenja s spleta (RUS).

Poučevanje v času enaindvajsetega stoletja se je preoblikovalo v novo obliko, ki vključuje vse več uporabe digitalnih komunikacijskih orodij v izobraževanju. Prav tako je prisotnih vse več različnih mrežnih aplikacij, ki skupaj s spreminjajočimi se značilnostmi, potrebami in zahtevami učencev tvorijo novo celoto. Danes je ena izmed pomembnih želja učencev tudi biti del aktivne učne izkušnje, ki je ob tem tudi socialna, participativna in podprta z IKT. Tako se kaže tudi vse večja potreba po podpori in spodbujanju učenčevega nadzora nad celotnim procesom izobraževanja [5]. Čeprav danes vse več uporabljamo različne platforme za e-učenje (te omogočajo vsakemu učencu osebni vpogled v izobraževanje), nekateri izražajo dvome ob itakšnem učenju. Sicer ne gre pozabiti, da se tradicionalni modele izobraževanja v spletnih okoljih prilagajajo, da je izkušnja učenja vendarle prijetna in v skladu s predpisi ter učnimi cilji. Zato je pomembno, kot navajajo tudi Green, Facer, Rudd, Dillon in Humphreys [5], da izobraževanje v digitalnih učnih okoljih tudi:

- zagotovi, da so učenci sposobni sprejemati premišljene izobraževalne odločitve,
- diverzificira in priznava različne oblike spretnosti in znanja,
- ustvarja raznolika učna okolja in
- vključuje oblike povratnih informacij in ocenjevanja, osredotočene na učenca [5].

Ob tem nikakor ne smemo pozabiti dati poudarka temu, kakšnim informacijam so učenci izpostavljeni (o čemer smo pisali že v uvodu). Zavedati se je treba, da se učenci lahko ob podpori tehnologije učijo na več načinov, a v osnovi je razlika, ali se bodo učili:

- vodeno, skupaj z učiteljem in bodo tako izpostavljeni tistim informacijam, ki jih je učitelj, praviloma pred poukom, tudi preveril,
- samostojno, z iskanjem informacij po različnih kanalih, zlasti po različnih dostopnih spletnih straneh ter programih in bodo morali učenci samostojno presojati, katere informacije so verodostojne in katere ne.

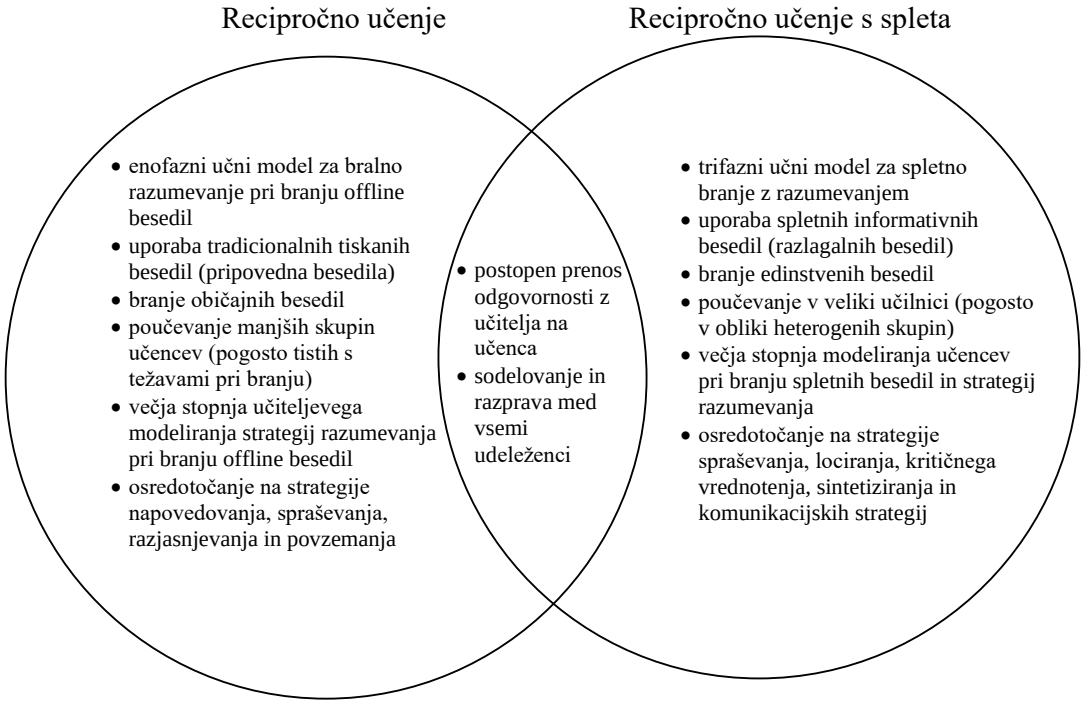
V okviru sodobnega z IKT podprtega učenja se lahko zato hitro zgodi, da so učenci podvrženi neresničnim oziroma lažnim informacijam. Slednje definiramo kot izmišljene oziroma neresnične informacije, ki posnemajo vsebino novičarskih medijev v obliki, ne pa v organizacijskem procesu ali namenu [7]. Kot posledica se lahko pojavi situacija, ko so učenci deležni zavajajočih informacij (neresničnih, nepopolnih) in dezinformacij (napačne informacije, ki se namerno širijo, da bi zavajale ljudi). Če je učenec deležen takšnih informacij, si bo hitro ustvaril napačno mnenje in stališče do obravnavane tematike, njegovo znanje bo nepopolno, lahko tudi povsem neustrezno [7]. A temu se je mogoče izogniti, če namenimo nekaj pozornosti kritičnemu mišljenju in učence spodbudimo k razmisleku, da je pomembno kritično presojati informacije, ki jih najdejo na spletu. Kako to vključiti v izobraževanje, predstavimo v nadaljevanju.

3 RECIPROČNO UČENJE S SPLETA (RUS)

Recipročno učenje odraža razkorak med dejansko stopnjo razvoja (samostojno reševanje problemov) in stopnjo potencialnega razvoja z reševanjem problemov pod vodstvom učitelja, ki prevzame vlogo mentorja ali moderatorja. V nekaterih primerih lahko tudi vrstniki prevzamejo vloga mentorja. Gre za vrsto poučevanja, ki temelji na konceptu vodene učne dejavnosti, ki se lahko uporablja med branjem in vključuje majhne skupine učencev, v katerih se učenci izmenično pogovarjajo z mentorjem o tem, kaj so prebrali. Recipročno učenje s spleta (RUS) je posebna vrsta recipročnega učenja, ki kot vir

namesto knjig uporablja svetovni splet in podatke, pridobljene s spleta, za pomoč učencem pri iskanju odgovorov na vprašanja in rešitev problemov. Izkušenost in postopno opuščanje odgovornosti učitelja pri uporabi metode pomagata učencem razviti strategije, ki postanejo samoregulirane in jih je mogoče prenesti na nove bralne naloge [6].

RUS je torej isto kot recipročno učenje, razlikuje se le v tem, da bralci poskušajo uporabiti svetovni splet za iskanje odgovorov na vprašanja, da bi razumeli in se učili [3]. Primerjava je prikazana na sliki 1, s pripisom *Primerjava recipročnega učenja in recipročnega učenja s spleta [6]*.Primerjava recipročnega učenja in recipročnega učenja s spleta *[6]*. [6].



Slika 1: Primerjava recipročnega učenja in recipročnega učenja s spleta [6].

Model RUS zajema tri stopnje, ki so predstavljene v tabeli 1 [6].

Faza 1: Pouk pod vodstvom učitelja	V prvi fazi so učenci deležni modeliranja, ki ga vodi učitelj. Ta vodi tudi razpravo in na glas predstavi razmišljanje ob delu. Delo praviloma poteka v obliki večje skupine. Učitelj ob tem spremlja ter opazuje napredek učencev, pri čemer uporablja kontrolni seznam TICA (Teacher Internet Comprehension to Adolescents) za prvo fazo. Ko ugotovi, da učenci razumejo delovanje modela, so pripravljeni na naslednjo fazo.
Faza 2: Skupno modeliranje spletnih raziskav in strategij razumevanja	Faza dve predstavlja premik od poučevanja učitelja k sodelovanju učencev v majhnih skupinah, pri čemer dejavnosti temeljijo na problemskem učenju ter so povezane s standardi in cilji učnega načrta ter spletnimi veščinami bralnega razumevanja. V tej fazi učenje postane manj strukturirano, učenci pridobijo znanje o spretnostih in strategijah spletnega branja, ključnega pomena pa je, da spodbujamo neodvisnost, kar pomeni, da učenci svoje

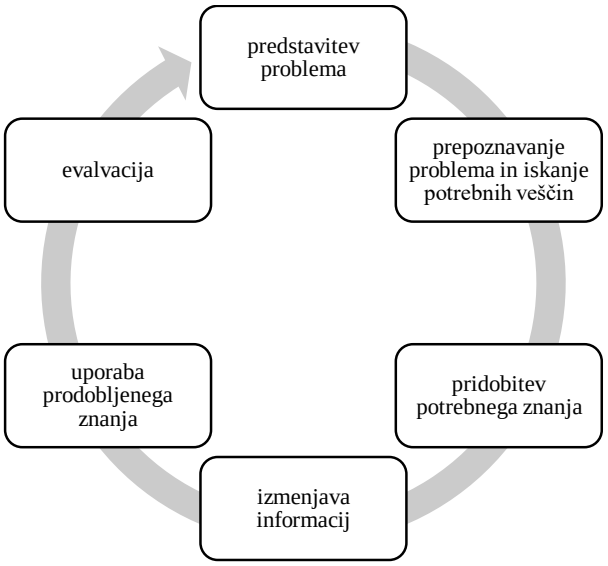
	vrstnike učijo strategij za spletno branje in učenje. Napredek se v tej fazi spremlja s kontrolnim seznamom TICA za drugo fazo.
Faza 3: Povpraševanje	Tretja faza vključuje poizvedovalno učenje v majhnih skupinah (izjemoma individualno). Učenci oblikujejo lastna raziskovalna vprašanja ali probleme, ki jih morajo rešiti. Cilj aktivnosti je tudi, da učenci svoje ugotovitve učinkovito sporočijo sošolcem preko raziskovalnih projektov. Vloga učencev se spremeni, ta postane aktivna, saj morajo učenci vedeti, kaj vprašati, kako ovrednotiti, sintetizirati in posredovati informacije v kratki in jedrnati obliki.

Tabela 1: Trije koraki metode RUS.

4 PROBLEMSKOST POUKA IN PROBLEMSKI POUK

Kot navaja Aberšek [1], mora sodobna šola omogočiti učencem več situacij, v katerih so podvrženi ustvarjalnejšemu mišljenju. Da to dosežemo, je treba učence spodbujati k bolj zahtevnim ravnem razmišljanja, med katere spadajo zlasti analiza, uporaba in ustvarjanje. Posebnost naštetih je tudi ta, da lahko pri njih razvijamo kritično mišljenje, ki je ključnega pomena, saj nam omogoča dobro podlago za reševanje različnih problemov, ki smo jih deležni v različnih situacijah v življenju [1]. Danes si učenci vse bolj želijo aktivne vloge pri učenju, zlasti takšne, ki so življenjsko usmerjene. To učencem omogoča, da se naučijo presojanja ter primerjanja novih spoznanj z lastnimi stališči in prepričanji. Kot navaja Aberšek [1], problemskost v osnovi zato uvrščamo v hevristično učno strategijo. Cilj slednje je učenje po načinu ustvarjanja, iskanja, odkrivanja ter raziskovanja. Seveda velja omeniti, da problemskost ni vsaka novost, temveč je problemskost tudi kompleksnost, protislovnost idr., zanjo pa je značilno, da je mnogoslojna in obojestranska [1]. Načeloma problemskost označujemo s tremi pojmi: problemska situacija, problem ter didaktična problemska naloga.

Iz problemskosti izhaja problemski pouk. Ideja problemskega poučevanja je, da učenci rešujejo odprti problem in hkrati pridobivajo nova znanja, pri čemer je problemska situacija, s katero se morajo soočiti, iz resničnega življenja. Učna metoda problemskega učenja spodbuja predvsem sodelovalno učenje, krepi interakcijo med učenci in spodbuja kritično vrednotenje. Učitelj zagotavlja podporo, usmerjanje in spremlja učni proces pri prizadevanju za problemsko usmerjeno poučevanje. To pomeni, da ima v razredu vlogo spremljevalca ali mentorja. Pri tem načinu poučevanja je velik poudarek na novo pridobljenem znanju, ki ga učenci uporabijo pri reševanju problema sami [1].

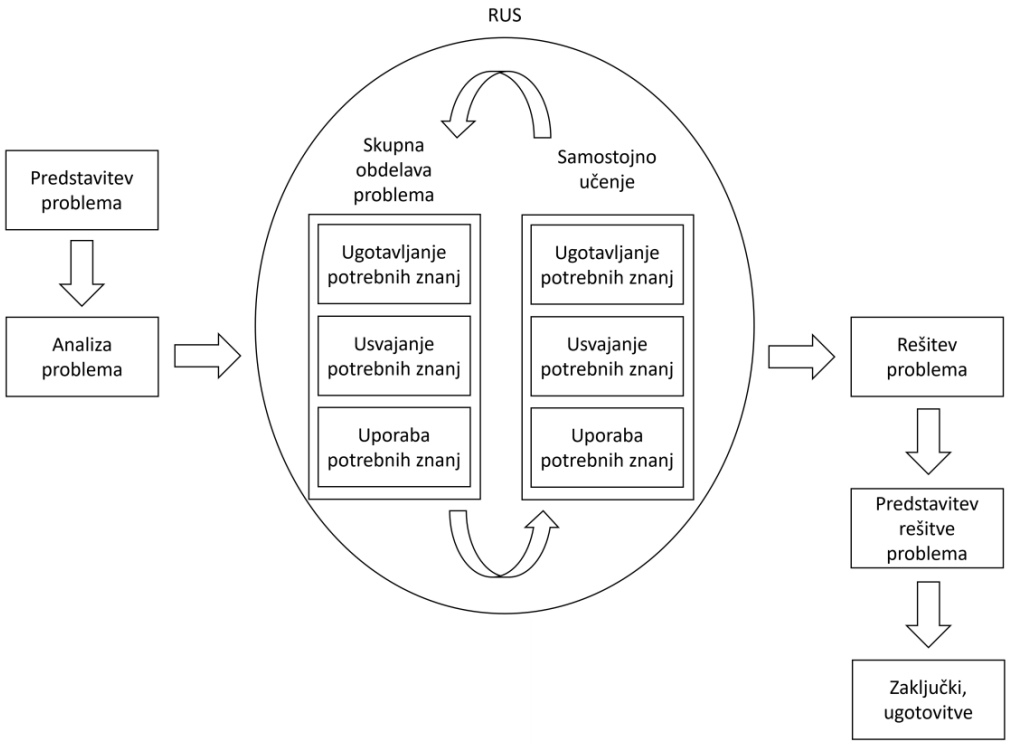


Slika 2: Koraki problemskega učenja [1].

Problemsko učenje poteka v šestih korakih, ki so prikazani na sliki 2. Najprej morajo učenci predstaviti problem, na katerem morajo delati. Nato učenci sami raziščejo ta problem in ugotovijo, katera znanja in informacije potrebujejo za rešitev problema. Nato začnejo iskati zahtevana znanja in informacije (bodisi po knjigah kot v našem primeru, pa tudi po metodi RUS). Nato si z raziskavo pridobljene informacije izmenjajo in s pridobljenim znanjem in informacijami predlagajo rešitev problema. Na koncu vse ocenijo [1].

6 POVEZAVA METODE RUS IN PROBLEMSKEGA POUKA

V enem izmed prejšnjih poglavij smo opisali tri faze metode RUS. Spomnimo, prva faza predstavlja delo pod vodstvom učitelja, druga faza predstavlja samostojno delo in delo v manjših skupinah, pri čemer so učenci kolikor se da samostojni in delajo ob različnih dejavnostih, tretja faza pa predstavlja situacijo, pri kateri učenci oblikujejo lastna raziskovalna vprašanja ali probleme, ki jih morajo rešiti. Opazimo lahko, da sta zlasti fazi 2 in 3 neposredno povezani s problemskim učenjem. V obeh fazah učenci namreč delajo ob različnih učnih dejavnostih, ki temeljijo na problemskem učenju, in so povezane s spletnimi veščinami bralnega razumevanja, ki jih zajema metoda RUS. Zato se nam zdi pomembno, da učenci spoznajo metodo RUS, preden so deležni problemskega učenja.



Slika 3: Kombinacija metode RUS in problemskega učenja. Vir: lasten.

Menimo, da je pri problemskem učenju v sodobnem z IKT tehnologijo podprtem učenju, potrebno kvalitetno načrtovanje dela in dober razmislek učitelja pred izvedbo problemsko osnovanega pouka. Ni namreč dovolj, da si zamislimo zgolj problemsko situacijo ali problem, ki ga morajo učenci rešiti, ampak je zahteva večja. Prva koraka problemskega pouka (predstavitev problema ter prepoznava problema in njegova analiza) nista tako podvržena kritičnemu mišljenju in razmišljanju kot nadaljnji koraki. Korak, kjer morajo učenci pridobiti ustrezno znanje in informacije, kako rešiti problem, je ključen. Tukaj so učenci praviloma prepuščeni (samostojnemu ali skupinskemu) iskanju informacij in znanj, v sodobnem izobraževalnem sistemu zlasti po svetovnem spletu. In prav tukaj je pomembno, da so sposobni kritično ovrednotiti pridobljene informacije ter izločiti tiste, ki niso relevantne. Da so to sposobni storiti, je potrebno, da so vešči takšnega počeja. Zato moramo učence pred tem korakom opolnomočiti, da so kritični in znajo presoditi, kaj je res in kaj ne, kaj je za rešitev problema koristno in kaj ne.

Ali so informacije, ki so jih pridobili, prave in koristne ali ne, je učencem težko presoditi, učitelj, ki pa je prevzel vlogo moderatorja, pa se težko osredotoča na vse podrobnosti reševanja problema, ki so jih ubrali učenci. Da se ne zgodi, da bi učenci usvojili napačna znanja, poskrbijo nadaljnji koraki problemskega učenja. Z izmenjavo informacij učenci preverijo, ali so prišli do enakih informacij in znanj, ki jih potrebujejo za reševanje problema (če problem rešujejo v več skupinah), prav tako jim uporaba pridobljenih informacij in znanja nakazuje, če gredo pri reševanju problema v pravo smer. Nenazadnje za končno potrditev ali ovrženje poskrbi evalvacija skupaj z vrednotenjem rezultatov. Če se izkaže, da je bila rešitev problema neuspešna, morajo učenci ponoviti nekatere korake problemskega pouka (začenši s korakom pridobivanja potrebnih informacij in znanj). In da se temu izognejo, zlasti na začetku, ko jim izkušnje problemskega učenja še manjkajo, jim metoda RUS pri tem pomaga, da že, ko prvič izvedejo ta korak, kritično ovrednotijo

informacije in znanje, do katerega so uspeli priti. Kako metoda RUS deluje v kombinaciji s problemskim učenjem, povzema slika 3.

7 ZAKLJUČEK

Svetovni splet je danes torej ključen vir informacij, ki ga učenci velikokrat uporabijo tudi pri učenju. A dejstvo je, da se vse pogosteje srečujemo s pojavljanjem neresničnih, nepopolnih in včasih tudi izmišljenih informacij. Največja težava nastopi, ko so tovrstnih informacij deležni učenci in jih uporabijo pri učenju. Posledično nas to pripelje do situacije, ko je učenčevo znanje v resnični nevarnosti, da je nepopolno ali celo napačno. Da bi se izognili podobnim situacijam je pomembno, da učence naučimo kritično presojati vire in informacije, ki jih pridobijo s spleta. Slednje nam omogoča metoda recipročnega učenja s spleta. Metodo je mogoče uporabiti pri pouku ob različnih učnih oblikah. V prispevku smo predstavili primer, kako lahko metodo recipročnega učenja s spleta uporabimo pri problemskem pouku. Če želimo problemsko metodo poučevanja združiti z metodo RUS, vidimo, da sta ti dve metodi združljivi v tretjem koraku problemskega poučevanja, ko morajo učenci iskati informacije. S prispevkom smo želeli učitelje spodbuditi k uporabi obeh metod pri poučevanju. Menimo, da bi sicer morali več pozornosti namenjati učnim virom, zlasti, če so ti na spletu oziroma so učenci večinoma časa podvrženi raziskovanju po spletu.

ZAHVALA

Članek je bil pripravljen kot povzetek vsebine izobraževanja učiteljev v LTT v Konyi, Turčija, na projektu Erasmus+ Modules and Applications for Increasing School Success of Students with the Innovative Digital School Model (Digi-School). Številka projekta: 2020-1-TR01-KA226-SCH-097822. Spletna stran projekta: http://digischoolproject.com/.

Literatura

[1] Aberšek, B. (2012). *Didaktika tehniškega izobraževanja med teorijo in prakso.* Didakta.

[2] Aberšek, B., Borstner, B., & Bregant , J. (2014). *Virtual Teacher: cognitive approach to e-learning material.* Newcastle upon Tyne: Cambridge Scholars Publishing.

[3] Alasmari N. (2021). Is Internet Reciprocal Teaching the Remedy for Saudi EFL Learners' Reading Difficulties During the Covid-19 Pandemic? *Journal of Education and e-Learning Research*, *8*(3), 324–332.

[4] Baylor, A., & Kim, Y. (2006). Educational Technology Research and Development. *A Social-Cognitive Framework for Pedagogical Agents as Learning Companions, 54*(6), 569-596.

[5] Green, Hannah & Facer, Keri & Rudd, Tim & Dillon, Patrick & Humphreys, Peter. (2005). *Futurelab: Personalisation and Digital Technologies*.

[6] Internet Reciprocal Teaching. *Teaching New Literacies.* [Online]. Pridobljeno: 5. 10. 2022. Dostopno na https://teachnewliteracies.wordpress.com/internet-reciprocal-teaching/

[7] Lazer D. M., Baum M. A., Benkler Y., Berinsky A. J., Greenhill K. M., Menczer F., Metzger M. J., Nyhan B., Pennycook G., Rothschild D., Schudson M., Sloman

S. A., Sunstein C. R., Thorson E. A., Watts D. J., & Zittrain J. L. (2018). The science of fake news. *Science,* 359(6380), 1094–1096. 10.1126/science.aao299829590025

[8] McLoughlin, C., & Lee, M. J. W. (2010). Personalised and self regulated learning in the Web 2.0 era: International exemplars of innovative pedagogy using social software. *Australasian Journal of Educational Technology,* 26(1). https://doi.org/10.14742/ajet.1100

[9] OECD (2022). *The Short and Winding Road to 2030: Measuring Distance to the SDG Targets.* OECD Publishing, https://doi.org/10.1787/af4b630d-en.

[10] United Nations. (b. d.). *Sustainable Development Goals.* [Online]. Pridobljeno: 1. 6. 2022. Dostopno na https://www.un.org/sustainabledevelopment/sustainable-development-goals/

STROKOVNI PRISPEVKI

Dianoia 7 (2023) 19-25

Nepričakovana spojina
Unexpected compound

Sara Gorenjak[1], Amalija Golobič[2], Brina Dojer[1]

[1] *Fakulteta za naravoslovje in matematiko, Koroška cesta 160, 2000 Maribor*

[2]*Fakulteta za kemijo in kemijsko tehnologijo, Večna pot 113, 1000 Ljubljana*

Povzetek

V prispevku poročamo o novi spojini, 3-aminopiridinijevem sulfatu hidratu, ki smo jo uspeli sintetizirati z reakcijo med železovim sulfatom heptahidratom, $FeSO_4 \cdot 7H_2O$, in 3-aminopiridinom, $C_5H_6N_2$, v metanolni raztopini. Karakterizirali smo jo z monokristalno rentgensko analizo in infrardečo spektroskopijo.

Spojina 3-aminopiridinijev sulfat hidrat, $6C_5H_7N_2^+ \cdot 3SO_4^{2-} \cdot 3H_2O$, kristalizira v triklinskem kristalnem sistemu (prostorska skupina *P*-1) s parametri osnovne celice: a = 7.0623(4) Å, b = 15.2121 (8) Å, c = 19.7611 (14) Å, α = 70.717(6) ˚, β = 80.024(6) ˚, γ = 82.834(5) ˚. Spojina tvori zapleten sistem intra- in intermolekularnih vodikovih vezi, veliko med njimi je bi- in trifurkalnih. Poleg vodikovih vezi strukturo dodatno stabilizirajo $\pi \cdots \pi$ interakcije.

Ključne besede: železov sulfat heptahidrat, aminopiridini, monokristalna rentgenska analiza, infrardeča spektroskopija, vodikove vezi

Abstract

In this paper we describe a new compound, 3-aminopyridine sulphate trihydrate, which we synzhesized by the reaction of iron sulphate heptahydrate, $FeSO_4 \cdot 7H_2O$, and 3-aminopyridine, $C_5H_6N_2$, in methanol solution. We characterized it by single crystal X-ray analysis and infrared spectroscopy.

The compound 3-aminopyridine sulphate trihydrate, $6C_5H_7N_2^+ \cdot 3SO_4^{2-} \cdot 3H_2O$, crystalizes in triclinic crystal system (space group *P*-1) with cell parameters: a = 7.0623(4) Å, b = 15.2121 (8) Å, c = 19.7611 (14) Å, α = 70.717(6) ˚, β = 80.024(6) ˚, γ = 82.834(5) ˚. The compound forms a complex system of intra- and intermolecular hydrogen bonds, many of which ared bi- and trifurcated. In addition to hydrogen bonds, the structure is stabilized by $\pi \cdots \pi$ interactions.

Key words: iron sulphate heptahydrate, aminopyridines, single crystal X-ray analysis, infrared spectroscopy , hydrogen bonds

1 UVOD

Naključno smo sintetizirali spojino 3-aminopiridinijev sulfat hidrat, $6C_5H_7N_2^+ \cdot 3SO_4^{2-} \cdot 3H_2O$. Naš namen je sprva bil pridobiti novo koordinacijsko spojino z reakcijo med železovim sulfatom heptahidratom in 3-aminopiridinom. Podobne sinteze smo izvajali že s solmi prehodnih kovin: kobaltom [4, 5], nikljem [2] in cinkom [3], produkti katerih so bile nove koordinacijske spojine. Znane so tudi koordinacijske spojine aminopiridinov z bakrom [8, 9, 10].

Pričakovali smo spojino, v kateri bi bili na železov centralni ion koordinirani ligandi, in sicer aminopiridin, sulfat in voda, vendar železovi ioni niso reagirali z aminopiridinom

kljub različnim načinom sinteze. Reakcije smo izvajali z običajnim mešanjem spojin v topilu pri sobni temperaturi, pod refluksom in s solvotermalno sintezo, pri kateri v zaprtem sistemu (avtoklavu) izvajamo dlje časa trajajoče reakcije pri povišani temperaturi in tlaku. Produkt je deloval kristalinično in obetajoče v smislu sinteze nove železove koordinacijske spojine, vendar se je kasneje izkazalo, da gre za nekovinsko spojino.

Spojine aminopiridin sulfatov smo zasledili tudi v literaturi, in sicer bis(4-aminopiridin) sulfat monohidrat [11], bis(2-aminopiridin) sulfat [7], 4-aminopiridin bisulfat anhidrat [6] in 4-aminopiridin sulfat hidrat [6]. Spojina, ki jo v nadaljevanju opisujemo, je nova. Zanimiva je predvsem zaradi velike asimetrične enote, ki vsebuje 6 aminopiridinijevih kationov, tri sulfatne anione in tri molekule vode. Posamezne enote se med seboj povezujejo s številnimi vodikovimi vezmi, strukturo pa dodatno stabilizirajo $\pi\cdots\pi$ interakcije med vzporednimi aromatskimi obroči kationov.

2 EKSPERIMENTALNI DEL

V 50 mL čašo smo zatehtali 0,1388 g (0,5 mmol) železovega sulfata heptahidrata, $FeSO_4 \cdot 7H_2O$, ter vsebino raztopili v 10 mL metanola. V drugo 50 mL čašo smo zatehtali 0,0943 g (1 mmol) 3-aminopiridina ter ga prav tako raztopili v 10 mL metanola. Raztopini železovega sulfata smo med mešanjem na magnetnem mešalu dodali raztopino 3-aminopiridina. Pripravljeno zmes smo prelili v bučko ter vpeli v pripravljeno aparaturo za sintezo z mešanjem pod refluksom (2 uri, 75 °C). Nastala je rjavo rdeča zmes, ki smo jo filtrirali in pridobili oranžen filtrat. Slednjega smo osušili v digestoriju. Ko je topilo izhlapelo, so se izločili kristali.
S preostalima sinteznima metodama nismo uspeli pridobiti ustreznih kristalov za monokristalno rentgensko analizo.

3 REZULTATI IN DISKUSIJA

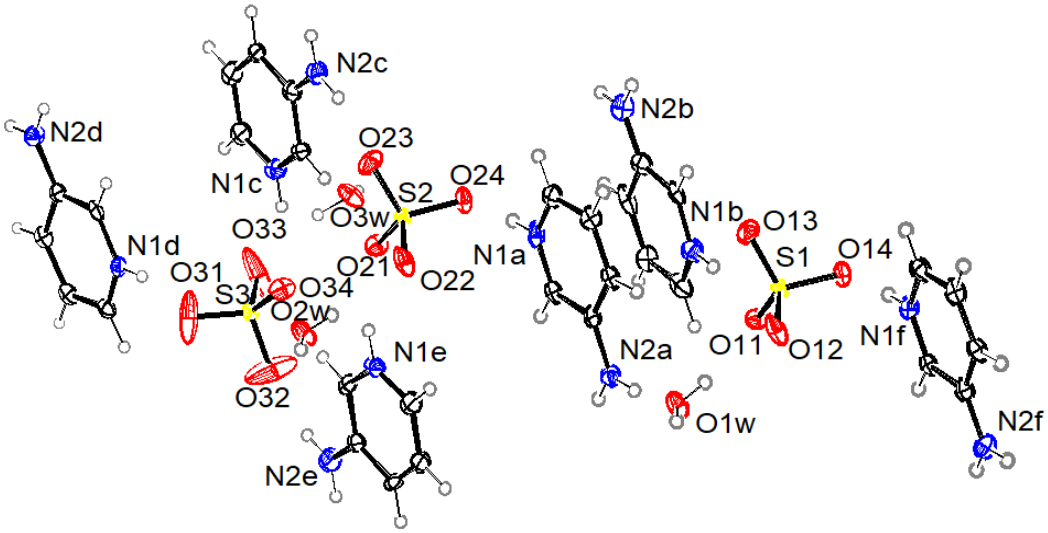
Nova spojina, katere struktura še ni objavljena, je kristalohidrat 3-aminopiridinijev sulfat s kemijsko formulo $6C_5H_7N_2^+ \cdot 3SO_4^{2-} \cdot 3H_2O$. Spojina kristalizira v triklinskem kristalnem sistemu s prostorsko skupino *P*-1 z naslednjimi parametri celice: a = 7,0623 (4) Å, b = 15,2121 (8) Å, c = 19,7611 (14) Å, α = 70,717 (6) °, β = 80,024 (6) °, γ = 82,834 (5)°. Kristalografski podatki so v tabeli 1.

Formula spojine	$6C_5H_7N_2^+ \cdot 3SO_4^{2-} \cdot 3H_2O$
Molska masa [g mol-1]	912,98
Barva kristala	svetlo oranžni
Kristalni sistem	triklinski
Prostorska skupina	*P*-1
Mrežne konstante	
a [Å]	7,0623 (4)
b [Å]	15,2121 (8)
c [Å]	19,7611 (14)
α [°]	70,717 (6)
β [°]	80,024 (6)

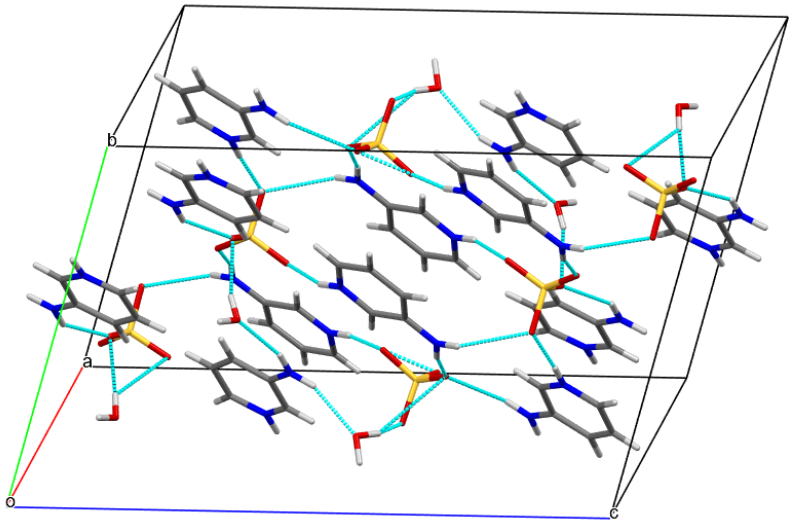
γ [°]	82,834 (5)
Volumen osnovne celice [Å^3], V	1968,37
Število formulskih enot v celici, Z	2
Izračunana rentgenografska gostota [g cm^{-3}]	1540
R – diferenčni faktor	0,0415

Tabela 1: Kristalografski podatki spojine.

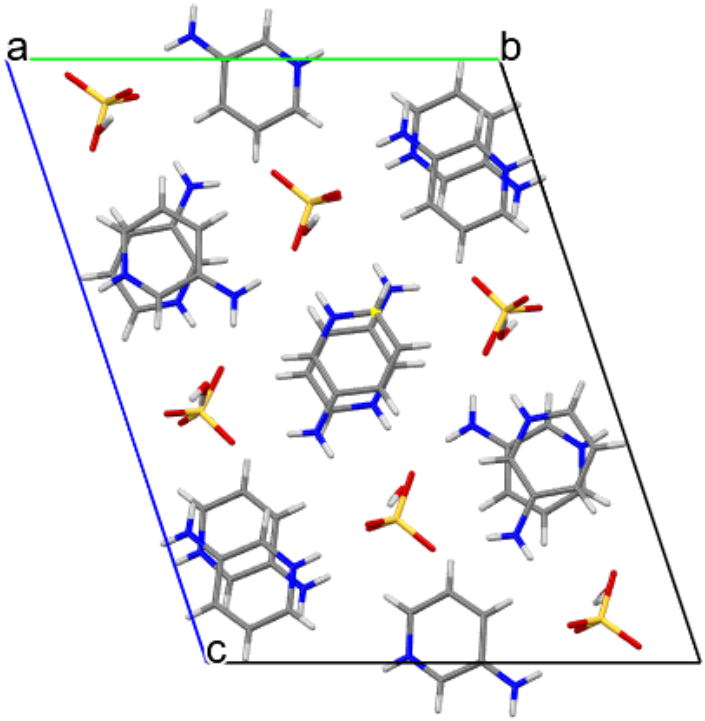
Struktura vsebuje šest aminopiridinijevih kationov, tri sulfatne anione in tri molekule vode. Razporejanje kationov, anionov in molekul vode v osnovni celici je prikazano na slikah 1 in 2.



Slika 1: Asimetrična enota z oznakami hetero atomov (N, O in S). Vir: lasten.



V kristalni strukturi so prisotne N–H···O in tudi O–H···O vodikove vezi. Geometrijski parametri so prikazani v tabeli 2. Najmočnejše vodikove vezi so tiste, ki jih donira obročni dušikov atom vsakega kationa (N1a, N1b, N1c, N1d, N1e in N1f), sprejema pa eden od kisikovih atomov iz sulfatnih anionov. Kontaktne razdalje N···O iz teh vodikovih vezi so namreč najkrajše (med 2,649 (4) in 2,727 (5) Å). Dušikov atom iz aminske skupine (N2a, N2b, N2c, N2d, N2e in N2f) donira vodikovo vez v vseh primerih preko obeh vodikovih atomov. Akceptorji teh vodikovih vezi so kisikovi atomi sulfatnih



Slika 2: V ove vezi.

Slika 3: Pogled na osnovno celico vzdolž roba a. V sredini vidimo prekrivanje vzporednih aromatskih obročev, med katerimi pride do π···π interakcij. Vir: lasten.

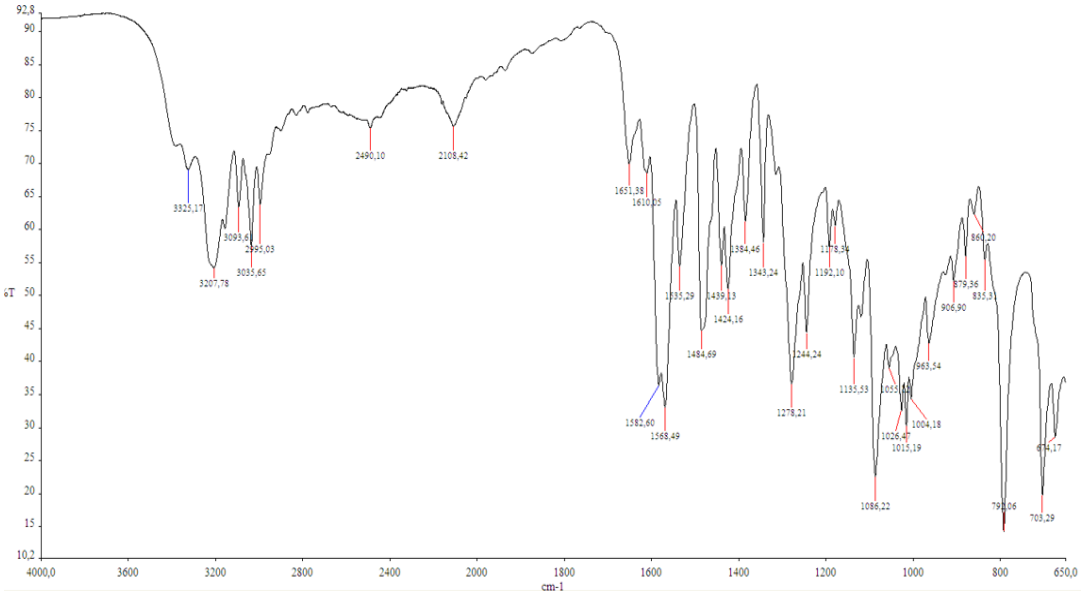
anionov in tisti iz molekul vode. Te vodikove vezi so šibkejše. Kontaktne razdalje N···O iz teh vodikovih vezi so namreč večje (med 2,914 (6) in 3,405 (8) Å). Molekulam vode donira eno vodikovo vez N2a in dve vodikovi vezi N2d atom. Preostale aminske skupine so povezane z vodikovimi vezmi s sulfatnimi ioni. Ta razlika v povezovanju aminske skupine kationov s sosedi je eden od razlogov, da je asimetrična enota tako velika. Vsaka molekula vode je vključena v tri vodikove vezi, in sicer je akceptor ene N–H···O vodikove vezi, ki jo donira dušikov atom iz aminske skupine, ter donor dveh O–H···O vodikovih vezi, katerih akceptor je vedno eden od kisikov iz sulfatnega iona. Vodikove vezi med molekulami vode in sulfatnimi ioni so zmerno močne. Kontaktne razdalje O···O so med 2,751 (6) in 2,881 (6) Å. Vsak sulfatni kisikov atom je akceptor vsaj ene vodikove vezi. Poleg vodikovih vezi strukturo dodatno stabilizirajo π···π interakcije med vzporednimi aromatskimi obroči kationov, kot je prikazano na sliki 3. Najkrajše razdalje med centri takih obročev so 3,468 (3), 3,512 (3) in 3,526 (3) Å.

D—H···A	D—H [Å]	H···A [Å]	D···A [Å]	D—H···A [°]
N1a—H1a1···O24	0,86	1,83	2,686 (4)	174
N1b—H1b1···O11	0,86	1,88	2,726 (5)	166
N1c—H1c1···O34	0,86	1,84	2,685 (5)	168
N1d—H1d1···O31	0,86	1,87	2,715 (5)	166
N1e—H1e1···O21	0,86	1,9	2,727 (5)	161
N1f—H1f1···O14	0,86	1,79	2,649 (4)	175
N2a—H2a1···O1w	0,86	2,14	2,963 (5)	161
N2a—H2a2···O32i	0,86	2,53	3,241 (7)	140
N2a—H2a2···O33i	0,86	2,58	3,405 (8)	163
N2b—H2b1···O22ii	0,86	2,33	3,118 (5)	152
N2b—H2b2···O12iii	0,86	2,61	3,388 (6)	151
N2b—H2b2···O13iii	0,86	2,34	3,102 (5)	148
N2c—H2c1···O23	0,86	2,13	2,970 (5)	164
N2c—H2c2···O14iv	0,86	2,44	3,229 (5)	153
N2d—H2d1···O2wv	0,86	2,14	2,975 (5)	163
N2d—H2d2···O3wvi	0,86	2,1	2,916 (5)	157
N2e—H2e1···O32	0,86	2,07	2,914 (6)	166
N2e—H2e2···O24i	0,86	2,5	3,264 (5)	149
N2f—H2f1···O12vii	0,86	2,23	3,010 (5)	150
N2f—H2f2···O23viii	0,86	2,34	3,102 (5)	147
O1w—H1w1···O14ix	0,88 (7)	1,98 (7)	2,847 (5)	171 (6)
O1w—H2w1···O12	0,80 (6)	2,02 (6)	2,815 (5)	174 (5)
O2w—H1w2···O24ix	0,89 (7)	1,94 (8)	2,821 (5)	172 (6)
O2w—H2w2···O22	0,78 (7)	2,07 (7)	2,800 (5)	157 (7)
O3w—H2w3···O31ii	0,86 (9)	2,14 (8)	2,881 (6)	143 (7)
O3w—H1w3···O33	0,87 (5)	1,92 (6)	2,751 (6)	159 (5)
N2*f*—H2*f*2···O23^{viii}	0,86	2,34	3,102 (5)	147
O1*w*—H1*w*1···O14^{ix}	0,88 (7)	1,98 (7)	2,847 (5)	171 (6)
O1*w*—H2*w*1···O12	0,80 (6)	2,02 (6)	2,815 (5)	174 (5)
O2*w*—H1*w*2···O24^{ix}	0,89 (7)	1,94 (8)	2,821 (5)	172 (6)
O2*w*—H2*w*2···O22	0,78 (7)	2,07 (7)	2,800 (5)	157 (7)
O3*w*—H2*w*3···O31^{ii}	0,86 (9)	2,14 (8)	2,881 (6)	143 (7)
O3*w*—H1*w*3···O33	0,87 (5)	1,92 (6)	2,751 (6)	159 (5)

Simetrijske kode: (i) −x+1, −y+1, −z+1; (ii) x+1, y, z; (iii) −x+2, −y+2, −z; (iv) x, y−1, z; (v) −x+1, −y, −z+1; (vi) −x+2, −y, −z+1; (vii) −x+1, −y+3, −z; (viii) x, y+1, z; (ix) x−1, y, z.

Tabela 2: Vodikove vezi – dolžine in koti (D= donor, A=akceptor).

Spojino smo karakterizirali tudi z infrardečo spektroskopijo. IR spekter prikazuje slika 4.



Slika 4: IR spekter spojine $6C_5H_7N_2^+ \cdot 3SO_4^{2-} \cdot 3H_2O$. Vir: lasten.

V vrhovih 3325 cm^{-1} in 3207 cm^{-1} spremljamo nihanja -NH_2 skupin. Trakovi, ki nakazujejo vzdolžno nihanje C-H skupin, se pojavijo okrog 3000 cm^{-1} (3094 cm^{-1}, 3036 cm^{-1} in 2994 cm^{-1}). Med 1650 – 1590 cm^{-1} pa opazimo deformacijska nihanja N-H skupin (1651, 1610, 1583, 1569 cm^{-1}). V območju okoli 1450 cm^{-1} se pojavijo nihanja aromatskih obročev (1439 cm^{-1}). Med 1430 in 1330 cm^{-1} se pojavljajo upogibna nihanja O-H skupin (1424 in 1384 cm^{-1}), prav tako se slednja pojavijo še v območju med 770 in 650 cm^{-1} (703, 674 cm^{-1}). V območju med 1340 cm^{-1} in 1200 cm^{-1} so vrhovi, ki nakazujejo raztezna nihanja aromatskih aminov (1343, 1278, 1244 cm^{-1}). Nihanja substituiranih piridinskih obročev so okoli 1000 cm^{-1}. Nihanja sulfatnih skupin se pojavijo med 1130 in 1080 cm^{-1} (1135 in 1086 cm^{-1}). C-H deformacije *meta*-substituiranih heteroaromatskih obročev lahko opazimo med 820 in 770 cm^{-1} (790 cm^{-1}) [1].

5 ZAKLJUČEK

Nepričakovano smo sintetizirali spojino 3-aminopiridinijev sulfat hidrat s kemijsko formulo $6C_5H_7N_2^+ \cdot 3SO_4^{2-} \cdot 3H_2O$. Podobne spojine z 2- in 4- aminopiridinom so znane, v prispevku omenjena spojina pa je nova. Analizirali in karakterizirali smo jo z monokristalno rentgensko analizo in infrardečo spektroskopijo. Kljub temu da smo pri sintezi uporabili železov sulfat heptahidrat, $FeSO_4 \cdot 7H_2O$, kristali niso vsebovali železovih ionov. Ti so po vsej verjetnosti ostali v praškastem delu produkta, ki ga z monokristalno rentgensko analizo ne moremo okarakterizirati.

Spojina je zanimiva predvsem zaradi prepleta vodikovih vezi. Slednje povezujejo aminopiridinijeve katione, sulfatne anione in molekule vode, ki sestavljajo asimetrično enoto, in prispevajo k stabilizaciji spojine. Poleg vodikovih vezi so v spojini tudi $\pi \cdots \pi$ interakcije med vzporednimi aromatskimi obroči kationov, ki strukturo dodatno stabilizirajo.

Literatura

[1] Bellamy, L. J. (1975). The Infra-red Spectra of Complex Molecules, Chapman and Hall, London.

[2] Dojer, B., Golobič, A., Jagličić, Z., Kristl, M., Drofenik, M. (2012). Two new nickel(II) carboxylates with 3- and 4-aminopyridine : syntheses, structures, and magnetic properties. *Monatshefte für Chemie*. vol. 143, no. 1, 73-78.

[3] Dojer, B., Pevec, A., Belaj, F., Kristl, M. (2015). Two new zinc(II) acetates with 3- and 4-aminopyridine: syntheses and structural properties. *Acta chimica slovenica.* vol. 62, no. 2, 312-318.

[4] Dojer, B., Pevec, A., Jagodič, M., Kristl, M., Drofenik, M. (2012). Three new cobalt(II) carboxylates with 2-, 3- and 4-aminopyridine: syntheses, structures and magnetic properties. *Inorganica Chimica Acta*, vol. 383, 98-104.

[5] Dojer, B., Pevec, A., Šegedin, P., Stropnik, Č., Kristl, M., Drofenik, M. (2010). Cobalt(II) coordination compounds with acetato and 2-aminopyridine ligands: Synthesis, characterisation, structures and magnetic properties of two polymorphic forms. *Inorganica Chimica Acta.* vol. 363, 1343-1347.

[6] Hursthouse, M.B., Montis, R., Niitsoo, L., Sarson, J., Threlfall, T.L., Asiri, A.M., Khan, S.A., Obaid, A.Y., Al-Harbi L.M. (2014). Anhydrates and/or hydrates in nitrate, sulphate and phosphate salts of 4-aminopyridine, (4-AP) and 3,4-diaminopyridine (3,4-DAP): the role of the water molecules in the hydrates,. *CrystEngComm.* Vol. 16, no.11, 2205-2219.

[7] Jebas, S. R., Balasubramanian, T., Peschar, R., Fraanje, J. (2006). Bis (2-aminopyridinium) sulfate. *Acta Crystallographica Section E: Structure Reports Online*. 62(7), o2606-o2607.

[8] Kozlevčar, B., Lah, N., Žlindra, D., Leban, I., Šegedin, P. (2001) Copper(II) benzoates and acetates with 2-aminopyridine. *Acta chimica slovenica*. vol. 48, no. 3, 363-374.

[9] Lah, N., Koller, J., Giester, G., Šegedin, P., Leban, I. (2002). Copper(II) carboxylates with 4-aminopyridine: neutral mononuclear structures, isomerism of compounds and a novel tetranuclear structure. *New journal of chemistry*. vol. 26, no. 7, 933-938.

[10] Lah, N., Šegedin, P., Leban, I. (2002). Crystal structures of two monomeric copper(II) carboxylates with 2-aminopyridine. *Structural chemistry*. vol. 13, no. 3/4, 357-360.

[11] Quah, C.K., FunH.-K., Isloorb, A.M., Isloorc, N. (2010) Bis(4-aminopyridinium) sulfate monohydrate, *Acta Crystallographica Section E: Structure Reports Online.* E66, o2250–o2251.

STROKOVNI PRISPEVKI

Dianoia 7 (2023) 27–39

Večnitne simulacije čustvenih stanj in modularni preizkusi v programskem jeziku Python

Multithreaded simulations of emotional states and unit testing in Python

Maša Galun, Ema Smolič, Drago Bokal

Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Koroška cesta 160, 2000 Maribor

Povzetek

Ob reševanju akademije Sekvenčni diagrami in simulacije čustvenih stanj v programskem jeziku Python avtorjev Dimič et al. spoznamo orodja za razvoj simulacij in izdelamo animacijo čustvenih stanj populacije posameznikov glede na različne nivoje strasti, vztrajnosti in učljivosti. Te koncepte sta v članku Faze psihološko optimalne učne izkušnje: model razporejanja časa na podlagi nalog predstavila Bokal in Steinbacher.
Naslednje naloge nas vodijo do razumevanja osnov večnitenja in modularnega testiranja. Ta nova pridobljena znanja prenesemo na programsko kodo iz članka Dimič et al.; s tem pospešimo izris animacije simulacij čustvenih stanj posameznika in obenem z modularnimi preizkusi preverimo pravilno delovanje programske kode.

Ključne besede: Večnitenje, simulacija, Python, modularni preizkusi, MPI

Abstract

By solving the exercises in the article Sequence Diagrams and Python Simulations of Emotional States by Dimič et al., we are acquainted with the tools to develop simulations and we create an animation of the emotional states in a population of individuals, based on different levels of passion, perseverance and learning ability. These concepts are introduced by Bokal and Steinbacher in the article Phases of a Psychologically Optimal Learning Experience: Task-based Time Allocation Model.
The following tasks lead us to master the basics of multithreading and unit testing. This new knowledge is then applied to the program code by Dimič et al.; thus speeding up the animation of an individual's emotional states, while also verifying the functioning of the program code by unit testing.

Key words: Multithreading, simulation, Python, unit testing, MPI

1 Uvod

V članku Faze psihološko optimalne učne izkušnje: model razporejanja časa na podlagi nalog [1] se spoznamo s simulacijami čustvenih stanj posameznika glede na njegovo stopnjo učljivosti, vztrajnosti in strasti pri učenju. V akademiji Sekvenčni diagrami in simulacije čustvenih stanj v programskem jeziku Python [3] se poučimo o osnovnih orodjih, ki jih potrebujemo za izdelavo simulacij. Naloge, ki sledijo, temeljijo na že rešenih nalogah iz članka [3], zato avtorji predpostavljamo, da ste te naloge že rešili. Cilj reševanja te akademije je, da nadgradimo znanje, ki smo ga že usvojili pri reševanju [3], in se spoznamo z večnitenjem ter modularnim testiranjem v programskem jeziku Python.

E-mail naslovi: galunmasa@gmail.com (Maša Galun), ema.smolic@gmail.com (Ema Smolič), drago.bokal@um.si (Drago Bokal)

Naloge so razdeljene v tri sklope. V prvem preizkusimo delovanje svojega programa z modularnimi preizkusi, v drugem se spoznamo z osnovami večnitenja, v tretjem to znanje prenesemo na simulacije čustvenih stanj.

2 Modularno testiranje

Modularno testiranje je metoda testiranja, ki jo uporabimo, ko se želimo prepričati, da naša programska koda deluje pravilno. V funkciji modularnega preizkusa poženemo testno kodo za več različnih parametrov oz. vrednosti in ugotovimo, če koda deluje, kot je bilo pričakovano [9].
V razvojnem okolju PyCharm v programskem jeziku Python za modularne preizkuse uporabljamo zaganjalnik pytest. Na želenem razredu ustvarimo modularni preizkus z ukazom `Ctrl+Shift+T` in odkljukamo, katere metode v razredu želimo testirati. Tako se ustvari nov razred z imenom `TestImeRazreda`. Za modularno testiranje iz knjižnice `unittest` vključimo razred `TestCase` [7].
Pri testiranju najpogosteje uporabimo metodi `TestCase.assertEqual()` in `TestCase.assertAlmostEqual()`, s katerima preverjamo vrednosti, katere ocenjujemo kot potencialno problematične. Metodi delujeta tako, da primerjata dejanske vrednosti s pričakovanimi in v primeru neenakosti sprožita izjemo `AssertionError`, ki jo okolje prepozna kot neuspel test [8].
Modularno testiranje nam tudi pomaga ugotoviti, pri katerih vrednostih program ne deluje tako, kot je pričakovano. Ker je to preverjanje avtomatizirano, ga lahko kadarkoli ponovimo. V praksi se tako testiranje ponavlja ob vsakem spreminjanju programske kode in s tem preveri, ali koda kljub spremembam deluje po pričakovanjih. S tem se zagotavlja nadzor nad spremembami, ki zgolj dodajo ustrezne nove funkcionalnosti, starih, preverjenih pa ne spreminjajo.
Ko se testiranje zaključi, se v konzoli izpiše, ali je bilo testiranje uspešno.

2.1 Naloge z namigi

Naloga 2.1 Naredite modularni preizkus na metodi `probabilityGiveUp()` za 100 000 oseb z naključno izbranimi vrednostmi strasti. Ostali parametri naj bodo ves čas enaki. Ugotovite, pri katerih vrednostih strasti metoda ne deluje pravilno.
Namig: Sproti izpisujte vrednosti strasti.

Naloga 2.2 Dopolnite metodo `probabilityGiveUp()` tako, da bo delovala tudi za te vrednosti strasti.
Namig: Do napake pride zaradi izračunane prevelike vrednosti parametra probability, ki je računalnik ni zmožen shraniti (`OverFlowError`).

Naloga 2.3 Podobno dopolnite metodo `probabilityFailure()`, da bo delovala pravilno.

Naloga 2.4 Z modularnim preizkusom preverite, ali metoda `probabilityFailure()` deluje pravilno za poljubno izbrane vrednosti.

Naloga 2.5 Naredite modularni preizkus na metodi `radius()` za simulacijo s parametri `dSize = 1.0` in `rCenter = 0.005`. Metodo preverite za koordinate (0.5, 0.5), (0,

0) in (1,1) ter poljubne koordinate po lastni izbiri.
Namig: Ali ste uporabili metodo `assertAlmostEqual()`?

Naloga 2.6 Za enake parametre simulacije naredite modularni preizkus metode `angle()` za koordinate (0.5, 0.5), (1.5, 0.5) in (0.5, 1.5) ter poljubne koordinate po lastni izbiri.

Naloga 2.7 Naredite modularni preizkus metode `emotion_from_angle()` za enake koordinate kot pri nalogi 2.6 in parametra `dSize = 1.0` in `rCenter = 0.005`.

Naloga 2.8 Naredite modularni preizkus metod `get_state(0, 0, 0, 10, 50)` in `get_state(0.9, 0.9, 0.9, 10, 50)`. Pri tem naj bosta parametra `dSize = 1.0` in `rCenter = 0.005`.

Naloga 2.9 Z modularnim preizkusom preverite, ali metoda `get_matrix()` res vrne matriko pravilne dimenzije.

2.2 Rešitve

Rešitev 2.1

```
import random
from unittest import TestCase
from Individual import Individual

class TestIndividual(TestCase):
  def test_probability_give_up(self):
    for i in range(100000):
      if i % 10000 == 0:
        print(i)
      r=random.random()
      per = Individual(numberOfTasks=10, l=0.5, p=0.5, r=r)
      print(r)
      per.probabilityGiveUp(1)
```

Opazimo, da metoda ne deluje pravilno za vrednosti strasti, ki se približujejo k 1, a niso enake 1.

Rešitev 2.2

```
  def probabilityGiveUp(self, index):
    if self.perseverance == 1:
      return 0
    elif self.perseverance == 0:
      return 1
    else:
      try:
        prob=1/(1+math.exp((-self.coefGSkill*self.overallSkill[index]
                        +self.coefGChallenge*self.overallChallenge[index]
                        -self.coefGExpected*self.expectedAbility(index)
                        +self.coefGOffset)/(1-self.perseverance)))
      except OverflowError:
        prob = float('inf')
      return prob
```

Rešitev 2.3

```
def probabilityFailure(self, index):
  if self.perseverance == 1:
    return 0
  elif self.perseverance == 0:
    return 1
  else:
    try:
      prob=1/(1+math.exp((-self.coefFSkill*self.overallSkill[index]
                          -self.coefFExpected*self.expectedAbility(index)
                          +self.coefFOffset)/(1-self.perseverance)))
    except OverflowError:
      prob = float('inf')
    return prob
```

Rešitev 2.4

```
def test_probability_failure(self):
  for i in range(100000):
    if i % 10000 == 0:
      print(i)
    r=random.random()
    per = Individual(numberOfTasks=10, l=0.5, p=0.5, r=r)
    print(r)
    per.probabilityFailure(1)
```

Rešitev 2.5

```
def test_radius(self):
  sim = Simulation()
  sim.dSize = 1.0
  sim.rCenter = 0.005
  self.assertEqual(sim.radius(0.5, 0.5), 0)
  self.assertAlmostEqual(sim.radius(0, 0), 0.7071067812)
  self.assertAlmostEqual(sim.radius(1, 1), 0.7071067812)
```

Rešitev 2.6

```
def test_angle(self):
  sim = Simulation()
  sim.dSize = 1.0
  sim.rCenter = 0.005
  self.assertEqual(sim.angle(0.5, 0.5), 0)
  self.assertAlmostEqual(sim.angle(0.5, 1.5), 1.5707963267948966)
  self.assertAlmostEqual(sim.angle(1.5, 0.5), 0)
```

Rešitev 2.7

```
def test_emotion_from_angle(self):
  sim = Simulation()
  sim.dSize = 1.0
  sim.rCenter = 0.005
  self.assertEqual(sim.emotion_from_angle(0.5, 0.5), sim.UNDECIDED)
  self.assertEqual(sim.emotion_from_angle(0.5, 1.5), sim.AROUSAL)
  self.assertEqual(sim.emotion_from_angle(1.5, 0.5), sim.CONTROL)
```

Rešitev 2.8

```
def test_get_state(self):
  sim = Simulation()
  sim.dSize = 1.0
  sim.rCenter = 0.005
  self.assertEqual(sim.get_state(0, 0, 0, 10, 50), (sim.WORRY, 100))
  self.assertEqual(sim.get_state(0.9, 0.9, 0.9, 10, 50), (sim.FLOW, 100))
```

Rešitev 2.9

```
def test_get_matrix(self):
  sim = Simulation()
  sim.dSize = 1.0
  sim.rCenter = 0.005
  matrika = sim.get_matrix(50, 10, 0.0)
  self.assertEqual(len(matrika), 11)
  for i in range(11):
    self.assertEqual(len(matrika[i]), 11)
```

3 Osnove večnitenja

Večnitenje je tehnika programiranja, katere najpomembnejša prednost je hitrost. Pospeši izvajanje programov, saj delo razdeli na več niti, tako da se hkrati uporablja več računalniških jeder [5]. En strukturiran proces se razdeli na več nalog, ki se lahko izvajajo neodvisno. Zato se naloge lahko izvedejo vzporedno na več jedrih, več procesorjih ali celo več računalnikih.
Pri delu s paralelnimi procesi bomo uporabljali vmesnik za izmenjavo sporočil oz. MPI [6]. Le-ta se uporablja v večih programskih jezikih in je standardni vmesnik za vzporedno sporočanje. Pri paralelnem izvajanju se sporočila lahko izmenjujejo na več načinov, in sicer sinhrono, blokirano, neblokirano ter kombinirano [10].
V naslednjih nalogah bomo sporočila pošiljali in prejemali blokirano.
Blokirana komunikacija poteka tako, da odgovor sporočila, ki ga prvega pošljemo, tudi prvega sprejmemo. V primeru, da so drugi odgovori že pripravljeni na sprejem, počakamo, da prispe prvi odgovor in ga uspešno sprejmemo. Šele nato v pravilnem vrstnem redu sprejmemo preostale odgovore. Ko prispejo vsi odgovori, se blokirano prejemanje zaključi in podatki so pripravljeni za nadaljnjo rabo [10].

Za naslednje naloge bomo potrebovali vzpostavljeno okolje za delo z MPI.

1. Prenesite in naložite datoteki:
   `https://www.microsoft.com/en-us/download/details.aspx?id=57467`
2. Nastavite okoljske spremenljivke s pomočjo povezave: `https://docs.oracle.com/en/database/oracle/r-enterprise/1.5.1/oread/creating-and-modifying-environment-variables-on-windows.html#GUID-DD6F9982-60D5-48F6-8270-A27EC53807D0`
   Treba je dodati sledeči mapi:

   - `C:\Program Files (x86)\Microsoft SDKs\MPI`
   - `C:\Program Files\Microsoft MPI\Bin`

3. Naložite paket mpi4py. Za dodatno pomoč obiščite: `https://mpi4py.readthedocs.io/en/stable/install.html`

Za zagon programa, ki ga želite pognati v večnitnem načinu, sledite naslednjim navodilom: Odprite mapo, kjer je vaš projekt. V naslovno vrstico vpišite `cmd`, tako se vam bo odprlo okno z ukaznim pozivnikom.
Vanjo vpišite: `mpiexec -n X python "CCC"`, pri čemer je X število želenih niti v vašem programu in CCC absolutno ime datoteke s Python kodo.
Ukaz `mpiexec` zažene program, ki mu ga podamo kot parameter v okolju MPI za večnitno delovanje. Predstavlja vmesnik med programom in informacijskim sistemom, ki Python programu, napisanemu z uporabo paketa mpi4py, omogoča večnitno izvedbo. Zadnji ukaz predstavlja pot do projekta, ki ga želite zagnati. Izberite `.py` datoteko in kopirajte njeno pot.

3.1 Naloge z namigi

Naloga 3.1 Iz knjižnice mpi4py vključite paket MPI. Vzpostavite okolje za delo z ukazom `MPI.COMM_WORLD` in pridobite informacijo o številu procesov z metodo `Get_rank()`.

Naloga 3.2 Izpišite `"Hello, World!"` najprej za eno nit, nato še za pet niti. Pri tem izpišite tudi indeks ranga, ki opravlja proces.

Naloga 3.3 Napišite program blokirane komunikacije med dvema procesoma. Proces z rangom 0 (nadzornik) naj pošlje sporočilo procesu z rangom 1 (delavec). Preverite tudi, da je delavec sporočilo prejel in se nanj odzval.

Naloga 3.4 Napišite program, ki izpiše poštevanko števil od 1 do 10. Pri tem naj nadzornik razdeli delo desetim delavcem.

Naloga 3.5 Program iz naloge 3.4 spremenite tako, da bo nadzornik delo razporedil poljubnemu številu delavcev.

3.2 Rešitve

Rešitev 3.1

```
from mpi4py import MPI

comm = MPI.COMM_WORLD
rank = comm.Get_rank()
```

Rešitev 3.2

```
print("{} : Hello, World!".format(rank))
```

Za rešitev z eno nitjo v okno vpišite: `mpiexec -n 1 python "CCC"`
V oknu se vam mora izpisati:

```
0 : Hello, World!
```

Za rešitev s petimi nitmi v okno vpišite: `mpiexec -n 5 python "CCC"`
V oknu se vam mora izpisati:

```
3 : Hello, World!
4 : Hello, World!
2 : Hello, World!
1 : Hello, World!
0 : Hello, World!
```

Pri čemer opazimo, da je zaporedje izpisanih rangov naključno.

Rešitev 3.3

```
if rank == 0:
  data = "message"
  print("rank 0 is sending a message to rank 1")
  comm.send(data, dest=1, tag=0)
  print("rank 0 successfully sent a message to rank 1")
elif rank == 1:
  print("rank 1 is receiving a message from rank 0")
  data = comm.recv(source=0, tag=0)
  print("rank 1successfully received a message from rank 0: ", data)
else:
  print("rank ", rank, "is not expecting anything")
```

Ukaz za zagon: `mpiexec -n 2 python "CCC"`

Rešitev 3.4

```
if rank == 0:
  for i in range(1, comm.size):
    comm.send(i, i, 0)
  for i in range(1, comm.size):
    data=comm.recv(None, i, 1)
    print(i, ":", data)
else:
  i = comm.recv(None,0,0)
  data = []
  for j in range(1, comm.size):
    data.append(i*j)
  comm.send(data, 0, 1)
```

Ukaz za zagon: `mpiexec -n 11 python "CCC"`

Rešitev 3.5

```
N = 10

if rank == 0:
  n = 1
  m = 1
  while 1:
    for i in range(1, comm.size):
      if n > N:
        break
      comm.send(False, i, 2)
      comm.send(n, i, 0)
      n += 1
    for i in range(1, comm.size):
      if m > N:
        break
      data=comm.recv(None, i, 1)
      print(i, ":", data)
      m += 1
    if m > N and n > N:
      break
  for i in range(1, comm.size):
    comm.send(True, i, 2)
else:
  while 1:
    end = comm.recv(None,0,2)
    if end == True:
      break
    i = comm.recv(None,0,0)
    data = []
    for j in range(1, N+1):
      data.append(i*j)
    comm.send(data, 0, 1)
```

Ukaz za zagon: `mpiexec -n X python "CCC"`, kjer je vrednost `X` število rangov. `X-1` je torej število delavcev.

4 Nadgradnja simulacije čustvenih stanj z večnitenjem

V tem poglavju bomo združili koncepte iz članka [1] z večnitenjem in s tem nadgradili programsko kodo iz [3] tako, da se lahko izvaja v večnitnem načinu. Hkrati bomo programski kodi iz [3] dodali različno zahtevne naloge. Do sedaj so bile vse naloge, ki jih je opravljal posameznik, enako zahtevne.
Za komunikacijo med procesi večnitenja bomo ponovno uporabili blokirano komunikacijo. V procesu nadzornik sodeluje z več delavci. Nadzornik bo delavcem razdelil delo in združil njihove rezultate v celoto. Sam ne bo opravljal nobenega dodatnega dela.
Cilj tega poglavja je, da s pomočjo večnitenja pohitrimo izris animacije čustvenih stanj iz [3].

4.1 Naloge z namigi

Naloga 4.1 V razredu za posameznika dopolnite metodo `__init__()` tako, da bodo naloge različno zahtevne. Stopnje zahtevnosti naj bodo cela števila med 1 in 10.
Namig: Ustvarite tudi polje, ki bo merilo trenutne stopnje zahtevnosti nalog, ki so na začetku vse enake 1.

Naloga 4.2 Dopolnite metodo `learn()` tako, da bo posameznik nalogo opravljal, dokler nad njo ne obupa ali jo uspešno zaključi. Ob uspešnem reševanju naj posameznik nadaljuje reševanje na višjem nivoju, ob neuspešnem naj reševanje ponovi na trenutnem nivoju. Če posameznik obupa, se naj reševanje te naloge predčasno zaključi in nadaljuje z reševanjem nove naloge.

Naloga 4.3 Razred `Simulation` spremenite tako, da bodo vse metode v njem definirane kot statične. Ustvarite nov razred `Animation` in metodi `color_matrices_and_animation` in `color_matrix` prenesite vanj.

Naloga 4.4 V razredu `Animation` napišite metodo `master(numberOFTasks,num_ind)`, ki naj deluje po principu blokiranega pošiljanja. Nadzornik naj delavcem pošlje polje oblike `[numberOFTasks, num_ind, l]` in od posameznega delavca prejme matriko čustvenih stanj, ki jo shrani v polje. Metoda vrne pridobljeno polje.
Namig: Pomagajte si z nalogama 3.4 in 3.5.

Naloga 4.5 V razredu `Animation` napišite metodo `worker()`, ki naj od metode `master(numberOFTasks, num_ind)` prejme polje `[numberOFTasks, num_ind, l]` in za njegove parametre izdela matriko čustvenih stanj. Le-to naj pošlje nadzorniku.

Naloga 4.6 Napišite statično metodo `animate(numberOfTasks, num_ind)`, ki za rang 0 (nadzornik) vrne matrike čustvenih stanj, pridobljene v metodi `master()`. Za ostale range naj kliče metodo `worker()`.
Namig: Pazite, da imate v konzoli navedeno pot, kjer se nahaja vaš projekt, saj se bo animacija shranila v to mapo.

Naloga 4.7 Napišite statično metodo `init_mpi(comm, numberOfTasks, num_ind)`, ki inicializira delo z MPI. Metoda naj s pomočjo metode `animate` naredi matrike čustvenih stanj in za rang 0 (nadzornik) iz teh matrik tudi izdela animacijo z metodo `color_matrices_and_animation`.

4.2 Rešitve

Rešitev 4.1

```
def __init__(self, numberOfTasks, l, p, r):
  self.numberOfTasks = numberOfTasks
  self.learning = l  # stopnja ucljivosti
  self.passion = p  # stopnja strasti
  self.perseverance = r  # stopnja vztrajnosti

  self.learningPeriod = 1000

  self.overallSkill = []
  self.overallChallenge = []

  self.overallLevel = []
  self.overallCurrLevel = []

  for i in range(numberOfTasks):
    self.overallSkill.append(np.random.uniform(0, 1))
    self.overallChallenge.append(np.random.uniform(0, 1))
    self.overallLevel.append(np.random.randint(1, 10))
    self.overallCurrLevel.append(1)
```

Rešitev 4.2

```
def learn(self):
  for i in range(self.learningPeriod):
    index = np.random.randint(0, self.numberOfTasks)
    if self.overallCurrLevel[index] <= self.overallLevel[index]:
      j = self.overallCurrLevel[index]
      while j <= self.overallLevel[index]:

        pf = self.probabilityFailure(index)
        pg = self.probabilityGiveUp(index)

        r = np.random.uniform(0, 1)

        if r < pg:
          self.giveup(index)
          break
        else:
          r = np.random.uniform(0, 1)
          if r < pf:
            self.failure(index)
          else:
            self.success(index)
            self.overallCurrLevel[index] += 1
            j += 1

      s = []  # skills
      c = []  # challenges
      for ind in range(self.numberOfTasks):
        s.append(self.overallSkill[ind])
        c.append(self.overallChallenge[ind])
      self.Hist_S.append(s)
      self.Hist_C.append(c)
```

Rešitev 4.4

```
comm = MPI.COMM_WORLD
rank = comm.Get_rank()

class Animation:
  comm = None
  rank = None

  @staticmethod
  def master(numberOFTasks,num_ind):
    matrices = []
    count_send = 1
    count_receive = 1
    l = 0.0
    while 1:
      for i in range(1, comm.size):
        if count_send > 11:
          break
        Animation.comm.send(False, i, 2)
        Animation.comm.send([numberOFTasks,num_ind, l], i, 0)
        l += 0.1
        count_send += 1
      for i in range(1, Animation.comm.size):
        if count_receive > 11:
          break
        matrix = Animation.comm.recv(None, i, 1)
        matrices.append(matrix)
        count_receive += 1
      if count_send > 11 and count_receive > 11:
        break
    for i in range(1, Animation.comm.size):
      Animation.comm.send(True, i, 2)
    return matrices
```

Rešitev 4.5

```
  @staticmethod
  def worker():
    while 1:
      end = Animation.comm.recv(None,0,2)
      if end:
        break
      arr = Animation.comm.recv(None, 0, 0)
      numberOfTasks = arr[0]
      num_ind = arr[1]
      l = arr[2]
      sim = Simulation()
      matrix = sim.get_matrix(numberOfTasks, num_ind, l)
      Animation.comm.send(matrix, 0, 1)
```

Rešitev 4.6

```
  @staticmethod
  def animate(numberOfTasks, num_ind):

    if Animation.rank == 0:
      matrices = Animation.master(numberOfTasks, num_ind)
      return matrices

    else:
      Animation.worker()
```

Rešitev 4.7

```
@staticmethod
def init_mpi(comm, numberOfTasks, num_ind):
  Animation.comm = comm
  Animation.rank = comm.Get_rank()

  matrices = Animation.animate(numberOfTasks, num_ind)

  if Animation.rank == 0:
    Animation.color_matrices_and_animation(matrices)
```

5 Zaključek

Skozi reševanje akademije smo nadgradili programsko kodo iz akademije Sekvenčni diagrami in simulacije čustvenih stanj v programskem jeziku Python [3] in vanjo vključili razlikovanje nalog glede na njihovo zahtevnost. Spoznali smo se z osnovami modularnega testiranja in večnitenja ter jih vključili v programsko kodo iz akademije [3]. Z večnitenjem smo tako pospešili izris animacije čustvenih stanj. Po rešeni akademiji smo zato še dodatno poglobili razumevanje konceptov iz članka Faze psihološko optimalne učne izkušnje: model razporejanja časa na podlagi nalog [1]. Ob reševanju smo spoznali osnove MPI komunikacije in na podlagi le-te utrdili razumevanje osnovnih konceptov večnitenja. Poučili smo se o prednostih uporabe modularnih preizkusov pri programiranju in z njimi preverili delovanje svoje programske kode.

Naslednji naraven korak v spoznavanju tehnologij, uporabnih v simulacijah z opisanimi matematičnimi modeli, je modularno testiranje programske kode, ki omogoča vzporedno izvajanje modelov. Ker paket `mpi4py` ni združljiv s paketom `unit test`, je za testiranje MPI kode potrebno posnemanje (emulacija) večnitenja v okolju Python, kar bi omogočalo zajem podatkov, potrebnih za modularno testiranje. Naredili smo prve korake v to smer: posnemanje večnitenja omogoča paket `mockmpi` [4].
Pri uporabi tega paketa pa smo naleteli na omejitve obstoječe implementacije, ki ne podpira vseh funkcionalnosti obstoječega MPI paketa. Težava je podrobneje opisana v komunikaciji z razvijalci paketa [2]. Ob prilagoditvi matematičnega modela komunikacije med posameznimi nitmi bi bil ta naslednji korak lahko izvedljiv; to bi bila hkrati zanimiva dopolnitev objavljenega prispevka in tudi koristna dopolnitev paketa `mockmpi`.

Literatura

[1] Bokal, D., Steinbacher, M. Phases of psychologically optimal learning experience: task-based time allocation model. Cent Eur J Oper Res 27 (2019), 863–885.

[2] Bokal D., Zuntz J., Jarvis M. (12. 7. 2022). TypeError: recv() takes 2 positional arguments but 4 were given[Forum objava].
Pridobljeno iz:`https://github.com/rmjarvis/MockMPI/issues/6`

[3] Dimic J., Lakner A., Potočan K., Bokal D. Sekvenčni diagrami in simulacije čustvenih stanj v programskem jeziku Python. Dianoia 5 (2021), 53–69.

[4] MockMPI 0.8.1. (2022).
Pridobljeno iz:`https://pypi.org/project/MockMPI/`

[5] Multithreading. (2022).
Pridobljeno iz:https://www.techopedia.com/definition/24297/multithreading-computer-architecture

[6] Open MPI: Open Source High Performance Computing. (2022).
Pridobljeno iz:https://www.open-mpi.org/

[7] Step 3. Test your first Python application. (2022).
Pridobljeno iz:https://www.jetbrains.com/help/pycharm/testing-your-first-python-application.html

[8] Unit testing framework. (2022).
Pridobljeno iz:https://docs.python.org/3/library/unittest.html#assert-methods

[9] Unit Testing – What is Its Importance in Software Testing?. (2020).
Pridobljeno iz:https://www.testingxperts.com/blog/unit-testing

[10] Vegi Kalamar A. (2018). Paralelni razveji in omeji algoritem BiqMac Solver (Magistrska naloga). Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Maribor.
Pridobljeno iz: https://dk.um.si/Dokument.php?id=129953&lang=slv

STROKOVNI PRISPEVKI

Dianoia 7 (2023) 41–54

Povabilo v prostore znanja
An invitation to knowledge spaces

Maša Galun, Drago Bokal

Univerza v Mariboru, Fakulteta za naravoslovje in matematiko, Koroška cesta 160, 2000 Maribor

Povzetek

V naslednjih nalogah na podatkovni strukturi sklad in vrsta pogledamo skozi prizmo učnih prostorov ter napišemo njuna razreda v programskem jeziku Python. Spoznamo se s pojmom učnih prostorov in s pomočjo znanja o podatkovnih strukturah definiramo lastne učne prostore. Vpeljemo pojem nekonsistentnega učnega prostora in na primeru podatkovnih struktur ugotovimo, kako nekonsistentne učne prostore narediti konsistentne. Nova znanja nato uporabimo pri risanju grafov učnih prostorov kot delnih kock. Pokažemo, da je graf vsakega učnega prostora tudi delna kocka.

Ključne besede: učni prostor, nekonsistentnost, delna kocka, sklad, vrsta

Abstract

In the following tasks, we study the learning spaces of data structures stack and queue. In addition, we present Python classes for the data structures. We introduce learning spaces and define learning spaces with the help of knowledge about data structures we gained. This leads us to introducing the concept of an inconsistent learning space while also studying how to make inconsistent learning spaces consistent. This knowledge is then applied to represent learning spaces as partial cubes. We show that the graph of every learning space is a partial cube.

Key words: learning space, inconsistency, partial cube, stack, queue

1 Uvod

Učni prostor je matematična struktura za modeliranje procesa učenja, spoznavanja novih konceptov [2]. Če evklidski prostori primarno izhajajo iz modeliranja fizičnega prostora in koncepta bližine oz. razdalje, so učni prostori namenjeni modeliranju intelektualnega prostora, znanj, ki jih posamezniki že posedujejo in jih lahko še pridobijo. Stanje v učnem prostoru, v katerem se posameznik nahaja, lahko le-ta pridobi s samorefleksijo svojega znanja in počutja pri uporabi tega znanja; objektivno pa je to stanje mogoče izmeriti s pomočjo preverjanja pravilnosti rešitev nalog, ki jih posameznik v nekem stanju znanja pravilno reši. Zaradi njihove uporabnosti smo se odločili, da tematiko približamo študentom in jim spoznavanje z učnimi prostori povezanih konceptov predstavimo z nalogami.

Naslednje naloge so razdeljene na tri sklope. V prvem spoznamo podatkovni strukturi sklad in vrsta ter v programskem jeziku Python napišemo lastna razreda za strukturi. Sklad in vrsta sta podatkovni strukturi, ki elemente shranjujeta v obliki seznama, vendar se bistveno razlikujeta v delovanju nekaterih razrednih metod. Definiramo abstraktno podatkovno strukturo koš, ki vsebuje vse skupne lastnosti sklada in vrste, in nato iz nje izpeljemo bodisi

E-mail naslovi: galunmasa@gmail.com (Maša Galun), drago.bokal@um.si (Drago Bokal)

sklad bodisi vrsto. Pogledamo tudi algoritem za iskanje v drevesu, v katerem si pomagamo s podatkovnima strukturama sklad in vrsta. V drugem razdelku se spoznamo s konceptom učnega prostora, ki nam pomaga razumeti, kako poteka učenje več med seboj povezanih konceptov in kako so le-ti odvisni eden od drugega. Definiramo učne prostore za sklad, vrsto in koš. Prav tako definiramo pojem nekonsistentnega učnega prostora, ki vsebuje znanja, ki se jih ni mogoče naučiti skupaj, in na primeru pokažemo, kako nekonsistentne učne prostore predelamo v konsistentne. V zadnjem poglavju se spoznamo z delnimi kockami in učne prostore iz drugega sklopa narišemo kot delne kocke. Dokažemo tudi, da je graf vsakega učnega prostora delna kocka.

2 Sklad, vrsta in koš

Sklad je urejeni seznam, kjer elemente vstavljamo samo na vrh seznama in imamo dostop le do elementa na vrhu [3].
Vrsta je urejeni seznam, kjer elemente vstavljamo na konec seznama in imamo dostop samo do elementa na začetku [3].

2.1 Naloge z namigi

Naloga 2.1 Napišite razred za podatkovno strukturo *sklad*. Pri tem definirajte naslednje operacije:

- `vstavi(podatek)`, ki na vrh sklada vstavi podatek.
- `naslednji()`, ki vrne element na vrhu sklada.
- `prazno()`, ki vrne `True`, če v skladu ni elementov in `False` sicer.
- `odstrani()`, ki odstrani element z vrha sklada.
- `izpis()`, ki izpiše vse elemente v skladu.

Namig: Pomagajte si z virom [3].

Naloga 2.2 Napišite razred za podatkovno strukturo *vrsta*. Pri tem definirajte naslednje operacije:

- `vstavi(podatek)`, ki na začetek vrste vstavi podatek.
- `naslednji()`, ki vrne element na začetku vrste.
- `prazno()`, ki vrne `True`, če v vrsti ni elementov in `False` sicer.
- `odstrani()`, ki odstrani element z začetka vrste.
- `izpis()`, ki izpiše vse elemente v vrsti.

Namig: Pomagajte si z virom [3].

Naloga 2.3 Napišite razred za novo podatkovno strukturo *koš,* ki bo vsebovala vse skupne lastnosti sklada in vrste.
Namig: Pomagajte si z rešitvama 2.1 in 2.2.

Naloga 2.4 Iz razreda podatkovne strukture *koš* izpeljite razred podatkovne strukture *sklad* tako, da podeduje vse lastnosti podatkovne strukture koš. Razredu dajte ime `VitekSklad`.
Namig: Operacije, ki jih koš ne definira, povzamemo po rešitvi naloge 2.1.

Naloga 2.5 Iz razreda podatkovne strukture *koš* izpeljite razred podatkovne strukture *vrsta* tako, da podeduje vse lastnosti podatkovne strukture koš. Razredu dajte ime `VitkaVrsta`.
Namig: Operacije, ki jih koš ne definira, povzamemo po rešitvi naloge 2.2.

Naloga 2.6 Napišite funkcijo, ki bo iz tekstovne datoteke prebrala matriko sosednosti in jo vrnila.

Naloga 2.7 Napišite funkcijo, ki bo za parameter prejela matriko in vrnila graf. *Graf* naj ima za vsako vozlišče shranjen indeks vozlišča, seznam njegovih sosedov in obiskanost (`True` ali `False`).

Naloga 2.8 Napišite funkcijo `iskanjeVDrevesu(D, koren, x, ds)`, ki v drevesu `D` s korenom `koren` poišče vozlišče `x`. Pri tem je `ds` bodisi vrsta bodisi sklad.

Naloga 2.9 Dokažite, da algoritem iz naloge 2.8 graf pregleda v širino, če je `ds` vrsta.
Namig: Dokažite naslednje trditve:

1. Preden v drevesu z vrsto obiščemo vozlišče na nivoju k, so bila vsa vozlišča na nivoju vsaj k dodana v vrsto. [6]
2. Če je ob preiskovanju drevesa v nekem trenutku prvo vozlišče v vrsti oddaljenosti k od korena, potem so vsa vozlišča v vrsti oddaljenosti k ali $k + 1$ od korena. Če so v vrsti vozlišča oddaljenosti k, potem so ta v vrsti pred poljubnim vozliščem oddaljenosti $k + 1$. [6]
3. Vsa vozlišča v drevesu, ki so od korena oddaljena k, bodo obiskana prej kot poljubno vozlišče z oddaljenostjo več ali enako $k + 1$.

Naloga 2.10 Dokažite, da algoritem iz naloge 2.8 graf pregleda v globino, če je `ds` sklad.
Namig: Dokažite naslednjo trditev:
Naj bosta v in u sosednji vozlišči v drevesu, za kateri velja, da je v za k oddaljeno od korena in u za $k + 1$ oddaljeno od korena. Potem bodo vsi sosedi vozlišča u s sklada odstranjeni pred vsemi sosedi vozlišča v.

2.2 Rešitve

Rešitev 2.1

```
class Sklad:
  def __init__(self):
    self.polje = []

  def vstavi(self, podatek):
    self.polje.append(podatek)

  def naslednji(self):
    if len(self.polje) == 0:
      return None
    return self.polje[-1]

  def prazno(self):
    if len(self.polje) == 0:
      return True
    return False

  def odstrani(self):
    if len(self.polje) != 0:
      self.polje.pop()

  def izpis(self):
    print(self.polje)
```

Rešitev 2.2

```
class Vrsta:
  def __init__(self):
    self.polje = []

  def vstavi(self, podatek):
    self.polje.append(podatek)

  def naslednji(self):
    if len(self.polje) == 0:
      return None
    return self.polje[0]

  def prazno(self):
    if len(self.polje) == 0:
      return True
    return False

  def odstrani(self):
    if len(self.polje) != 0:
      for i in range(len(self.polje)-1):
        self.polje[i] = self.polje[i+1]
      self.polje.pop()

  def izpis(self):
    print(self.polje)
```

Rešitev 2.3

```
class Kos:
  def __init__(self):
    self.polje = []

  def vstavi(self, podatek):
    self.polje.append(podatek)

  def naslednji(self):
    return None

  def prazno(self):
    if len(self.polje) == 0:
      return True
    return False

  def odstrani(self):
    pass

  def izpis(self):
    print(self.polje)
```

Rešitev 2.4

```
from Kos import Kos

class VitekSklad(Kos):

  def naslednji(self): # vrh
    if len(self.polje) == 0:
      return None
    return self.polje[-1]

  def odstrani(self):
    if len(self.polje) != 0:
      self.polje.pop()
```

Rešitev 2.5

```
from Kos import Kos

class VitkaVrsta(Kos):
  def __init__(self):
    super().__init__()

  def naslednji(self): # zacetek
    if len(self.polje) == 0:
      return None
    return self.polje[0]

  def odstrani(self):
    if len(self.polje) != 0:
      for i in range(len(self.polje)-1):
        self.polje[i] = self.polje[i+1]
      self.polje.pop()
```

Rešitev 2.6

```
def preberi(str):
  mat = []
  f = open(str, 'r')
  for line in f:
    a = [int(i) for i in
        line.strip().split(" ")]
    mat.append(a)
  return mat
```

Rešitev 2.7

```
def ustvariGraf(mat):
  D = {}
  for i in range(len(mat)):
    sosedi = []
    for j in range(len(mat)):
      if mat[i][j] == 1:
        sosedi.append(j)
    vozlisce = [sosedi, False]
    D[i] = vozlisce
  return D
```

Rešitev 2.8

```
from VitkaVrsta import VitkaVrsta
from VitekSklad import VitekSklad
from Vrsta import Vrsta
from Sklad import Sklad

def iskanjeVDrevesu(D, koren, x, ds):
  if isinstance(ds, Vrsta):
    ds = IzpeljanaVrsta()
  else:
    ds = IzpeljanSklad()

  ds.vstavi(koren)
  while not ds.prazno():
    v = ds.naslednji()
    print(v)
    ds.odstrani()
    if D[v][1] == True:
      continue
    D[v][1] = True
    if v == x:
      return v
    for i in D[v][0]:
      if D[i][1] == False:
        ds.vstavi(i)
```

Rešitev 2.9

Izrek 1. *Preden z algoritmom iz rešitve 2.8 v drevesu z vrsto obiščemo vozlišče na nivoju k, so bila vsa vozlišča na nivoju vsaj k dodana v vrsto. [6]*

Dokaz: Recimo, da to ni res. Poglejmo prvi korak algoritma, ko trditev iz izreka ne velja. Obstajata torej vozlišče u na nivoju k, ki je obiskano, in vozlišče v na nivoju k' ($k' \leq k$) , ki še ni bilo dodano v vrsto. Naj bo u' zadnje vozlišče, ki je bilo obiskano, preden je bilo vozlišče u dodano v vrsto. Vozlišče u' je torej na nivoju $k-1$, saj sicer to ni prvi tak korak. Poglejmo vozlišče v', ki je sosednje z v in je na nivoju $k'-1$. To vozlišče obstaja, ker

obstaja najkrajša pot od korena do v in ker vemo, da vozlišče v ni koren, saj je bilo vozlišče u v vrsto dodano pred njim.
Ker velja $k' \leq k$, iz tega sledi $k' - 1 \leq k - 1$. Torej je bil v' dodan v vrsto, preden je bil u' obiskan.
Časovno zaporedje je torej:

1. v' je dodan v vrsto,
2. u' je obiskan,
3. u je dodan v vrsto,

Ker je v' dodan v vrsto pred u, to pomeni (zaradi lastnosti vrste), da je bil v' obiskan pred u. Ker je bil v' obiskan, smo v dodali v vrsto (saj sta vozlišči sosednji). To pomeni, da je bil v dodan v vrsto, preden je bil u obiskan. Protislovje, ki potrdi izrek. □

Izrek 2. *Če je ob preiskovanju drevesa z algoritmom iz naloge 2.8 v nekem trenutku prvo vozlišče v vrsti oddaljenosti k od korena, potem so vsa vozlišča v vrsti oddaljenosti k ali $k + 1$ od korena. Če so v vrsti vozlišča oddaljenosti k, potem so ta v vrsti pred poljubnim vozliščem oddaljenosti $k + 1$. [6]*

Dokaz: Prejšnji izrek pove, da so bila vsa vozlišča z oddaljenostjo k dodana v vrsto, preden je bilo katero od vozlišč z oddaljenostjo k obiskano oz. odstranjeno iz vrste. Torej so vsa vozlišča z oddaljenostjo k v vrsti pred vozlišči z oddaljenostjo $k + 1$, saj je vozlišče z oddaljenostjo $k + 1$ lahko dodano v vrsto tedaj, ko je bilo katero od vozlišč z oddaljenostjo k odstranjeno iz vrste.
Prav tako so vsa vozlišča z oddaljenostjo $k - 1$ bila dodana v vrsto pred poljubnim vozliščem z oddaljenostojo k. Zato vozlišča z oddaljenostjo $k - 1$ niso več v vrsti, ko je prvi element v vrsti oddaljenosti k od korena.
Tako bodo tudi vsa vozlišča oddaljenosti več kot $k+1$ dodana v vrsto, ko bodo vsa vozlišča oddaljenosti k že odstranjena iz vrste. □

Izrek 3. *Vsa vozlišča v drevesu, ki so od korena oddaljena k, bodo z algoritmom iz naloge 2.8 obiskana prej kot poljubno vozlišče z oddaljenostjo več ali enako $k + 1$.*

Dokaz: Ker naš algoritem deluje tako, da vozlišče doda v vrsto le, če to še ni bilo pregledano, bo vsak element v vrsto dodan samo enkrat. Izrek 2 pove, da bodo vsa vozlišča oddaljenosti k v vrsti pred poljubnim vozliščem z oddaljenosti $k + 1$ in da bodo takrat v vrsti le vozlišča teh oddaljenosti. Algoritem vozlišče obišče, ko je le-to prvo v vrsti. Ker so vsa vozlišča oddaljenosti k pred vozlišči oddaljenosti $k + 1$, bodo vozlišča oddaljenosti k obiskana prej kot vozlišča oddaljenosti $k+1$. Vozlišča oddaljenosti več kot $k+1$ po izreku 2 takrat še sploh niso bila dodana v vrsto, torej niso bila obiskana. □

Rešitev 2.10

Izrek 4. *Naj bosta v in u sosednji vozlišči v drevesu, za kateri velja, da je v za k oddaljeno od korena in u za $k + 1$ oddaljeno od korena. Potem bodo vsi sosedi vozlišča u s sklada odstranjeni pred vsemi sosedi vozlišča v.*

Dokaz: Poglejmo trenutek, ko smo obiskali vozlišče v. Označili smo ga kot obiskanega in v sklad dodali vse njegove sosede. Med njimi je tudi vozlišče u, ki naj bo brez škode za splošnost na vrhu sklada. Ker je u na vrhu sklada, ga obiščemo in v sklad dodamo vse njegove sosede. Opazimo, da so bili sosedi vozlišča u v sklad dodani za sosedi vozlišča v, od katerih je vsaj u še vedno v skladu. Ker ukaz `naslednji` od sklada iz rešitve naloge 2.1 vrne najprej zadnji vstavljeni podatek, bodo sosedi vozlišča u iz sklada odstranjeni prej kot u in morebitni preostali sosedi vozlišča v, kar pomeni, da bodo tudi prej obiskani.
Če pa je neko vozlišče sosed tako u kot v, pa bo dvakrat (morda tudi večkrat, odvisno od preostalih sosedov) dodano v sklad, najprej kot sosed v in potem kot sosed u. Prvič, ko se pojavi na vrhu sklada, bo obiskano in kot tako označeno. Vsakič naslednjič, ko bo na vrhu sklada, pa bo že označeno kot obiskano in bo zato zgolj odstranjeno z vrha sklada. □

3 Učni prostori

Definicija 1. [1]
Struktura znanja $(Q, \mathscr{K})$ je urejen par množice Q in družine $\mathscr{K}$ podmnožic množice Q, kjer je vsak element množice Q znanje in vsaka množica iz $\mathscr{K}$ stanje učenja znanj iz Q. Velja torej: $\mathscr{K} \subseteq \mathcal{P}(Q)$. Zahtevamo še, da velja $\emptyset \in \mathscr{K}$.

Definicija 2. [2]
Učni prostor $(Q, \mathscr{K})$ je struktura znanja, za katero velja:

1. DOSTOPNOST. $\forall K \in \mathscr{K} : K \neq \emptyset \Rightarrow \exists q \in K : K \backslash \{q\} \in \mathscr{K}$
   Stanje K dosežemo, če se naučimo znanje q v stanju $K \backslash \{q\}$.
2. KONSISTENTNOST UČENJA. $\forall K \in \mathscr{K}, \forall q, r \in Q : K \cup \{q\} \in \mathscr{K} \wedge K \cup \{r\} \in \mathscr{K} \Rightarrow K \cup \{q, r\} \in \mathscr{K}$
   Če se iz nekega stanja K lahko naučimo tako znanje q kot tudi znanje r, potem se lahko iz stanja K naučimo obe znanji.

3.1 Naloge z namigi

Naloga 3.1 Naj bo $\mathscr{A}$ množica množic, ki vsebuje $\emptyset$. Operacijo $* : \mathbb{Z}_2 \times \mathscr{A} \to \mathscr{A}$ za poljubno množico A definiramo kot: $x * A = \emptyset$ za $x = 0$ in $x * A = A$ za $x = 1$.
Z uporabo operacije $*$ v eni vrstici zapiši potenčno množico množice $A = \{1, 2, 3, 4, 5\}$.

Naloga 3.2 Napišite Python funkcijo, ki za parameter prejme strukturo znanja in zanjo preveri, ali je učni prostor.
Namig: Python zahteva, da pri delu z množicami množic uporabimo metodo `frozenset()`.

Naloga 3.3 Definirajte učni prostor za podatkovno strukturo *koš*.
Namig: S pomočjo prejšnje naloge preverite, ali je to res učni prostor.

Naloga 3.4 Definirajte učni prostor za podatkovno strukturo *sklad*.
Namig: Dopolnite učni prostor strukture *koš*.

Naloga 3.5 Definirajte učni prostor za podatkovno strukturo *vrsta*.
Namig: Dopolnite učni prostor strukture *koš*.

Naloga 3.6 Razmislite, kako bi definirali pojem nekonsistentnega učnega prostora. S pomočjo te definicije definirajte nekonsistentni učni prostor za podatkovni strukturi *sklad* in *vrsta*.
Namig: Nekonsistentni učni prostor omogoča učenje na podoben način kot konsistentni učni prostor. Od konsistentnega se razlikuje v tem, da v določenih primerih preprečuje učenje nekonsistentnih znanj. Na primer, operacij vrste in sklada iz uvodnega razdelka nalog se ne moremo naučiti skupaj, saj ima ista oznaka v vrsti in skladu različen pomen. Podobno ne moremo v istem učnem prostoru obravnavati 5. aksioma evklidske geometrije, sferične geometrije in hiperbolične geometrije. [5]

Naloga 3.7 Definirajte konsistentni učni prostor za podatkovne strukture *koš, sklad* in *vrsta*.
Namig: Metodi *odstrani* in *naslednji* naj bosta dva ločena elementa za podatkovni strukturi *sklad* in *vrsta*.

3.2 Rešitve

Rešitev 3.1
$\mathcal{P}(A) = \{x_1 * \{1\} \cup x_2 * \{2\} \cup x_3 * \{3\} \cup x_4 * \{4\} \cup x_5 * \{5\}; x_i = 0, 1, i = 1, 2, 3, 4, 5\}$

Rešitev 3.2

```
def dostopnost(Q, K):
  for i in K:
    ustreza = False
    if len(i) == 0 or len(i) == 1 or i == frozenset():
      continue
    for j in i:
      A = i.difference(frozenset({j}))
      if A in K:
        ustreza = True
        break
    if ustreza == False:
      return False
  return True

def konsistentnost(Q, K):
  for i in K:
    A = set()
    B = Q.difference(i)
    for q in B:
      A.add(q)
      for r in B.difference(A):
        if i.union(frozenset({q})) in K and i.union(frozenset({r})) in K:
          if i.union(frozenset({q}).union(frozenset({r}))) not in K:
            print(i, q, r)
            return False
  return True

def jeUcniProstor(Q, K):
  if dostopnost(Q, K) and konsistentnost(Q, K) and (frozenset() in K):
    return True
  return False
```

Rešitev 3.3
$A_0 = \{\}$
$A_1 = \{\text{podatek}\}$
$A_2 = \{\text{podatkovni tip}\} \cup A_1$
$A_3 = \{\text{seznam}\} \cup A_2$
$A_4 = \{\text{Boolean}\} \cup A_2$
$A_5 = \{\text{True}\} \cup A_4$
$A_6 = \{\text{False}\} \cup A_4$
$A_7 = \{\text{True}\} \cup A_6$
$A_{i+4} = \{\text{seznam}\} \cup A_i, i = 4, 5, 6, 7$
$A_{12} = \{\text{koš}\} \cup A_{11}$
$A_{12+x_{pi}+2x_{po}+4x_v} = A_{12} \cup x_{pi} * \{\text{pripravi}\} \cup x_{po} * \{\text{prazno}\} \cup x_v * \{\text{vstavi}\},$
$A_{19} = \{\text{podatek, podatkovni tip, seznam, Boolean, True, False, koš, pripravi, prazno, vstavi}\}$

Učni prostor: $(A_{19}, \{A_0, A_1, \ldots, A_{19}\})$

Rešitev 3.4
Množice $A_0 - A_{19}$ so enake kot v rešitvi 3.3.
$A_{20} = \{\text{sklad}\} \cup A_{19}$
$A_{21} = \{\text{odstrani}\} \cup A_{20}$
$A_{22} = \{\text{naslednji}\} \cup A_{20}$
$A_{23} = \{\text{odstrani}\} \cup A_{22}$

Učni prostor: $(A_{23}, \{A_0, A_1, \ldots, A_{23}\})$

Rešitev 3.5
Množice $A_0 - A_{19}$ so enake kot v rešitvi 3.3.
$A_{20} = \{\text{vrsta}\} \cup A_{19}$
$A_{21} = \{\text{odstrani}\} \cup A_{20}$
$A_{22} = \{\text{naslednji}\} \cup A_{20}$
$A_{23} = \{\text{odstrani}\} \cup A_{22}$

Učni prostor: $(A_{23}, \{A_0, A_1, \ldots, A_{23}\})$

Rešitev 3.6

Definicija 3. Struktura $(Q, \mathscr{K}, \mathscr{C})$ je *nekonsistentni učni prostor*, če velja:

1. $(Q, \mathscr{K})$ je struktura znanja.
2. DOSTOPNOST. Definicija enaka kot v navadnem učnem prostoru.
3. POGOJNA KONSITENTNOST UČENJA. $\forall K \in \mathscr{K}, \forall q, r \in Q : K \cup \{q\} \in \mathscr{K} \wedge K \cup \{r\} \subseteq \mathscr{K} \wedge \{q, r\} \notin \mathscr{C} \Rightarrow K \cup \{q, r\} \in \mathscr{K}$
   Če se lahko iz stanja K naučimo q in r in q, r nista nekonsistentna, potem se lahko naučimo obe znanji.

$\mathscr{C} \subseteq \mathcal{P}(Q)$ je množica nekonsistentnih elementov.

$\mathscr{C} = \{\{$vrsta.odstrani, sklad.odstrani$\}, \{$vrsta.naslednji, sklad.naslednji$\}\}$
Množice $A_0 - A_{22}$ so enake kot v rešitvi 3.4.
$A_{23} = \{$vrsta$\} \cup A_{19}$
$A_{24} = \{$odstrani$\} \cup A_{20}$
$A_{25} = \{$naslednji$\} \cup A_{20}$
$A_{26} = \{$odstrani$\} \cup A_{22}$
$A_{27} = A_{22} \cup A_{26}$

Učni prostor: $(A_{27}, \{A_0, A_1, \ldots, A_{26}\}, \mathscr{C})$

Rešitev 3.7
Množice $A_0 - A_{19}$ so enake kot v rešitvi 3.3.
$A_{19+x_v+2x_{so}+4x_z+8x_{vo}} = A_{19} \cup x_v * \{$vstavi$\} \cup x_{so} * \{$sklad.odstrani$\} \cup x_z * \{$naslednji$\} \cup x_{vo} * \{$vrsta.odstrani$\}$

Učni prostor: $(A_{34}, \{A_0, A_1, \ldots, A_{34}\})$

4 Delne kocke

Definicija 4. *Graf strukture znanja G* je definiran z $V(G) = \mathscr{K}$ in $E(G) = \{(u, v) | u, v \in \mathscr{K} \ \& \ \exists q \in Q : v = u \cup \{q\}\}$.

Izrek 5. *[4] Delna kocka je graf, katerega vozlišča lahko označimo z enako dolgimi nizi bitov tako, da je razdalja med poljubnima vozliščema grafa enaka vsoti različnih bitov teh vozlišč.*

4.1 Naloge

Naloga 4.1 Poiščite definicijo hiperkocke kot grafa.

Naloga 4.2 Poiščite definicijo delne kocke.

Naloga 4.3 Naj bo $\mathbb{K} = (Q, \mathscr{K})$ učni prostor. Dokažite, da je potem graf od $\mathbb{K}$ delna kocka.

Naloga 4.4 Narišite delno kocko za učni prostor podatkovne strukture koš iz naloge 3.3.

Naloga 4.5 Narišite delno kocko za učni prostor podatkovnih struktur sklad in vrsta iz nalog 3.4 in 3.5.

Naloga 4.6 Narišite delno kocko za nekonsistentni učni prostor iz naloge 3.6.

Naloga 4.7 Pokažite, da sta grafa učnih prostorov podatkovnih struktur sklad in vrsta iz naloge 4.5 izomorfna. V katerih povezavah se ustrezna učna prostora razlikujeta?

4.2 Rešitve

Rešitev 4.1

Definicija 5. [1] *n-razsežna hiperkocka* je graf na množici $\{0,1\}^n$, v katerem sta poljubni vozlišči povezani natanko tedaj, ko se razlikujeta v natanko eni poziciji.
n-razsežna hiperkocka se imenuje tudi n-kocka.

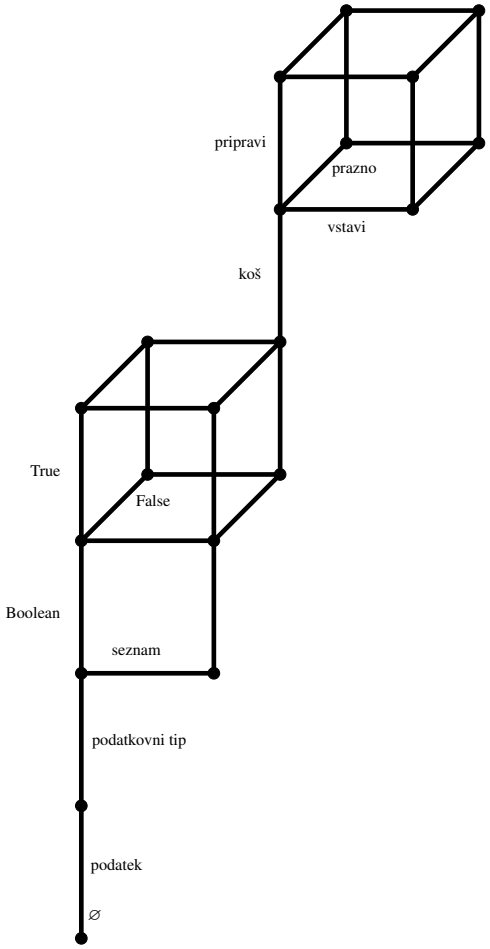
Rešitev 4.2

Definicija 6. [4] *Delna kocka* je graf, ki je izometrično izomorfen podgrafu hiperkocke.

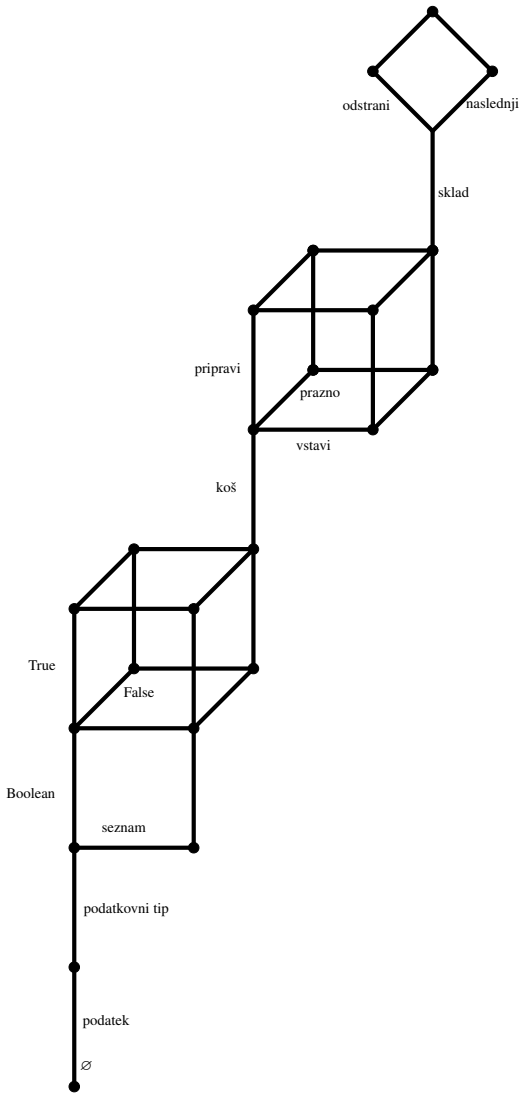
Rešitev 4.3

Naj bo G graf učnega prostora $\mathbb{K}$ in $Q = \{q_1, q_2, \ldots, q_n\}$. Velja torej $n = |Q|$. Vsakemu vozlišču grafa G bomo priredili število, sestavljeno iz n bitov.
Vsako vozlišče v grafu G predstavlja neko stanje iz $\mathscr{K}$. Za poljubno stanje torej velja, da bodisi vsebuje element q_i, bodisi ga ne vsebuje. Če vozlišče vsebuje znanje q_i, bo i-ti bit tega vozlišča 1, sicer pa 0.
Tako bo torej stanje $\emptyset$ sestavljeno iz samih ničel, vsakemu naslednjemu vozlišču pa spremenimo i-ti bit, ko v vozlišče pridemo iz že označenega stanja s spremembo i-tega znanja.
Opazimo, da bodo števila vozlišč na razdalji k od prazne množice sestavljena iz k enic in $n-k$ ničel. Prav tako se bosta poljubni sosednji vozlišči razlikovali v natanko enem bitu.
Če dokažemo, da je razdalja med poljubnima dvema vozliščema v grafu G enaka številu različnih bitov teh vozlišč, smo dokazali, da je G delna kocka.
Naj bosta $A, B \in \mathscr{K}$ poljubni vozlišči. Dokazujemo, da je $d(A,B) = |A \oplus B|$. Dokažimo, da je vsota različnih bitov vozlišč A in B enaka $|A \oplus B|$.
Po izreku 4.2.1 iz [1] velja, da v $\mathbb{K}$ za vsaka $A, B \in \mathscr{K}$ obstajata verigi stanj, tako da velja:
$A \subseteq A \cup \{b_1\} \subseteq A \cup \{b_1, b_2\} \subseteq \cdots \subseteq A \cup \{b_1, b_2, \ldots, b_k\} = A \cup B$
$B \subseteq B \cup \{a_1\} \subseteq B \cup \{a_1, a_2\} \subseteq \cdots \subseteq B \cup \{a_1, a_2, \ldots, a_l\} = A \cup B$
Pri tem velja $k = |B \backslash A|$ in $l = |A \backslash B|$.
Da pridemo iz stanja A v stanje $A \cup B$, torej spremenimo k bitov. Pri prehodu iz stanja $A \cup B$ v stanje B pa spremenimo l bitov. Skupaj smo spremnili $k + l$ bitov, ki so paroma različni. Če bi med stanjema A in B obstajala krajša pot, potem bi na eni povezavi spremenili dva bita hkrati, kar je v protislovju z definicijo grafa.

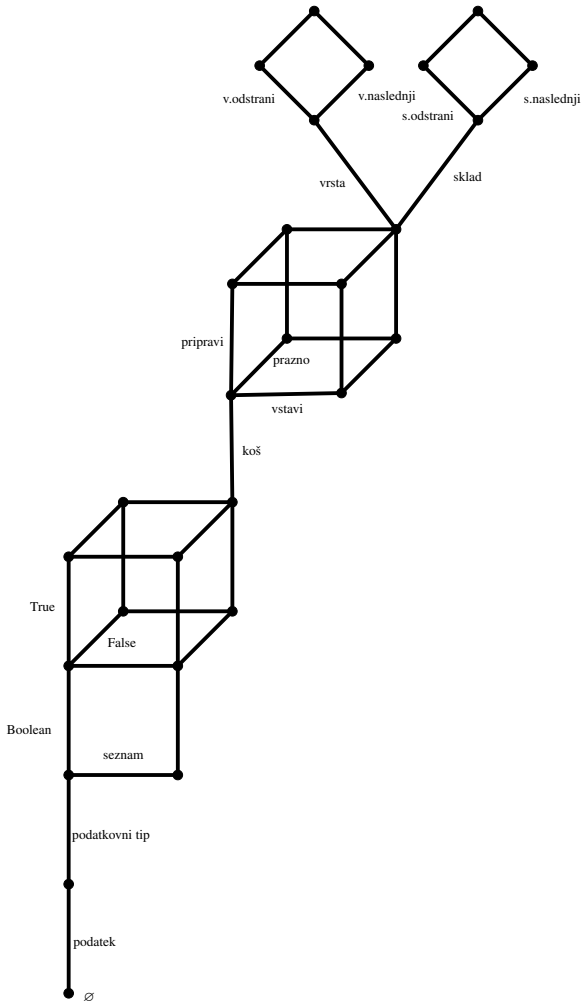
Rešitev 4.4



Rešitev 4.5



Rešitev 4.6



Rešitev 4.7
Naj bo $\mathbb{K} = (Q, \mathscr{K})$ učni prostor strukture sklad in $\mathbb{K}' = (Q', \mathscr{K}')$ učni prostor strukture vrsta.
Definirajmo preslikavo $f : Q \to Q'$:

$$f(q) = \begin{cases} \text{vrsta.naslednji; če } q = \text{sklad.naslednji} \\ \text{vrsta.odstrani; če } q = \text{sklad.odstrani} \\ q; \text{ sicer} \end{cases}$$

Očitno je, da je f bijekcija. Hitro se da preveriti tudi, da je preslikava F: $\mathscr{K} \to \mathscr{K}' = \{f(k)|k \in K\}$ bijekcija, ki slika vozlišča delne kocke vrste v vozlišča delne kocke sklada in ohranja povezanost. Zato je preslikava F izomorfizem grafa učnega prostora vrste v graf učnega prostora sklada. □
Grafa učnih prostorov se razlikujeta v tistih povezavah, ki so označena z znanji, na katerih preslikava f ni identiteta, torej na povezavah, označenih z vrsta.odstrani in vrsta.naslednji.

5 Zaključek

Z reševanjem nalog smo utrdili znanje o podatkovnih strukturah sklad in vrsta ter napisali lastna razreda zanju. Definirali smo novo abstraktno podatkovno strukturo koš, iz katere smo nato izpeljali sklad in vrsto. Napisali smo tudi algoritem za iskanje v drevesu in v njem uporabili podatkovni strukturi sklad in vrsta. Poučili smo se o strukturah znanja in posebnih primerih le-teh – učnih prostorih. Spoznali smo, kakšne pogoje mora izpolnjevati struktura znanja, da je učni prostor, in definirali učne prostore za sklad, vrsto in koš. Vpeljali smo pojem nekonsistentnega učnega prostora in ga prikazali na primerih podatkovnih struktur, obenem pa ugotovili, kako iz nekonsistentnih učnih prostorov zgradimo konsistentne učne prostore. Poučili smo se o delnih kockah in njihovi povezavi z učnimi prostori. Videli smo, da je graf vsakega učnega prostora tudi delna kocka. Učne prostore podatkovnih struktur smo narisali kot delne kocke.

Predstavljeni sklopi nalog so lahko uporabna podpora študentom, saj si ob učenju za nek predmet oz. izpit lahko študent na podlagi konceptov nariše učni prostor predmeta in s tem še bolj poglobljeno ugotovi, kako so med seboj povezani posamezni deli in koncepti določenega predmeta. Tak pregled nad predmetom služi kot zemljevid učenja: omogoča spremljanje napredka, usmerjanje in vodenje po stanjih znanja. V pomoč je lahko tudi učitelju, profesorju, ki v takem modelu lažje vidi iz katerega stanja znanj so študenti prišli in katera znanja obvladajo.

Odprt problem, ki se pojavi pri raziskovanju učnih prostorov, je nekonsistentni učni prostor evklidske, sferične in hiperbolične geometrije, ki imajo vsaka svoj aksiom o vzporednici skozi dano točko (5. aksiom). V istem učnem prostoru teh aksiomov ne moremo obravnavati skupaj, zato bi bilo zanimivo videti, kakšni so učni prostori izrekov v geometrijah, ki na videz delujejo zelo podobno, a se zaradi 5. aksioma bistveno razlikujejo v dokazu.

Literatura

[1] Eppstein D., Falmange J.C., Ovchinnikov S., *Media Theory*, first ed., Springer, New York, 2008.

[2] Falmange J.C., Doignon J.P., *Learning spaces*, third ed., Springer, New York, 2011.

[3] Kozak J., *Podatkovne strukture in algoritmi*, second ed., DMFA, Ljubljana, 1997.

[4] Partial cube. (2022).
Pridobljeno iz:https://en.wikipedia.org/wiki/Partial_cube

[5] The Three Geometries. (2022).
Pridobljeno iz:https://mathstat.slu.edu/escher/index.php/The_Three_Geometries

[6] University of California San Diego. (2022). BFS Properties[Video]. Coursera.
https://www.coursera.org/lecture/algorithms-on-graphs/proof-of-correctness-8aYP7?redirectTo=%2Flearn%2Falgorithms-on-graphs%3Faction%3Denroll

STROKOVNI PRISPEVKI

Dianoia 7 (2023) 55-67

Application of Binary protocol in testing blood samples

Uporaba binarnega protokola pri testiranju vzorcev krvi

Lara Prijatelj

Univerza v Ljubljani, Filozofska fakulteta, Aškerčeva cesta 2, 1000 Ljubljana, Slovenija

Abstract

The article deals with methods for the detection of rare diseases in blood samples with presumption that one or two samples in a population of a certain number of individuals are infected. Generally, irregularities in the blood samples are detected by examining the blood of each sample separately. The article aims to determine whether there is a method that would minimalize the number of tests that must be carried out in a population to detect the infected sample(s). Initially, three methods of weighing coins, a model for blood samples, are presented. Secondly, the task focuses on blood samples and a method called the Binary Protocol for rare abnormalities. The relationship between the expected value of the infected samples and the number of tests required to detect the infected sample, or more samples is observed. By using Binary Protocol, the number of tests needed to find the infected sample is reduced.

Key words: Binary protocol, blood samples, weighing, expected value

Povzetek

Članek obravnava metode za odkrivanje redkih bolezni v vzorcih krvi s predpostavko, da je v populaciji določenega števila posameznikov okužen en ali dva vzorca. Na splošno se nepravilnosti v vzorcih krvi odkrivajo s pregledom krvi vsakega vzorca posebej. Namen članka je ugotoviti, ali obstaja metoda, ki bi zmanjšala število testov, ki jih je treba opraviti v populaciji, da bi odkrili okužene vzorce. Na začetku so predstavljene tri metode tehtanja kovancev, ki so model za vzorce krvi. V nadaljevanju se naloga osredotoča na vzorce krvi in metodo, imenovano binarni protokol za redke nepravilnosti. Opazovana je povezava med pričakovano vrednostjo okuženih vzorcev in številom testov, potrebnih za odkrivanje okuženega vzorca ali več vzorcev. Z uporabo binarnega protokola se zmanjša število testov, potrebnih za odkrivanje okuženega vzorca.

Ključne besede: Binarni protokol, krvni vzorci, tehtanje, povprečna vrednost

1 INTRODUCTION

Early recognition and diagnosis of infectious diseases are of great importance as they prevent further spread of disease and enable successful treatment. This can be achieved by testing blood samples of infants. The question that arises is how to discern if the infant has a virus or a disorder. I assumed that diagnosis is made by withdrawing blood of infants and testing it for irregularities. "The presence of a particular substance in the blood would indicate the presence of the condition and vice versa" [1].

The general procedure to detect irregularities in the blood of infants is to examine the blood of each infant separately. If there are 100 infants, 100 tests of blood samples are conducted to ascertain which infants are healthy and which infected.

E-naslov: lara.prijatelj00@gmail.com (Lara Prijatelj)

Binary protocol presents a different approach to test blood samples by reducing the number of tests needed to find the infected sample. The general binary-splitting algorithm performs a binary search on groups that have tested positive. It is a simple algorithm that finds a single fault in the maximum number of tests corresponding to the lower bound of the information [7].
My attempt is to minimalize the number of tests that must be carried out in a sample of n infants in a population, to detect abnormalities. A sample is considered a subset of the population or a selection of the individuals from the population [4].
Before working with blood samples I will address the problem from a different perspective, by weighing a number of identical coins of same shape, size, colour and weight. However, one coin is forged and weighs more than other coins. The goal is to identify the forged by weighing the coins.
Weighing means observing left (L) and right (R) position of the scale when a coin is placed on each side. If the coins have the same weight, the scale is in balance. It is in a horizontal plane and does not lean [3]. If a coin on L has greater weight than coin on R, the scale leans to the left, indicating that the forged is on the left side and vice versa.
How many numbers of weighings must be performed to find a forged among a series of n coins if there is exactly one forged coin? Three different methods of dispensing coins are inspected.

1. Method: Maximum number of weighing the first coin with all the others

There are $n = 2, 3, 4, 5, \ldots$ coins, the position of the forged coin is unknown. Assuming that the forged is in the first two places in the series of n coins, the number of weighings is the smallest, equal to 1. The maximum number of weighings is necessary if the forged is in the last place in the series of n coins. In an example of $n = 3$, the first coin is weighed with the second coin. If the first coin is heavier than the second it is forged and vice versa. If both coins weigh the same, the third remaining coin is forged. The number of weighing is presented in Table 1.

Number of coins [n]	2	3	4	5	...	$\mathbf{n \geq 3}$
Number of weighing	1	1	2	3		$\mathbf{n - 2}$

Table 1: Maximum number of weighings needed to detect a forged coin in relation to number of coins

The maximum number of weighings needed to find the forged among all coins for $n \geq 3$ is $n - 2$. The number of weighings is always smaller than the number of coins in a series.

2. Method: dividing coins into pairs and weighing the pairs against each other

There are $n = 2, 3, 4, 5, \ldots$ coins, the position of the forged in the series of n coins is unknown. Two examples for even and odd numbered series are presented.

Four coins are split into two pairs of two. The forged coin is selected at random. Coins from the first pair are weighted against each other. Because the scale is in balance, the forged is not located in the first pair, weighing is continued. Coins from the second pair are weighted against each other. Since the scale is not in balance but leans to the left, the forged is found as the third in the series of four coins. In this example two weighings are needed to locate the forged coin. If the latter was located in the first pair, one weighing would be needed to find it.

Five coins are split into two pairs of two and a remaining coin. The forged coin is selected at random. Coins of the first pair are weighted against each other. Because the scale is in balance, the forged must be located either in the second pair or is the remaining coin. Coins of the second pair are weighted against each other, the scale is in balance. Therefore, the remaining coin is forged. Two weighings are needed to find the forged in this example and if it would be located in the second pair. If the forged coin is located in the first pair, one weighing would be needed to find it.
The number of weighings depends on the location of the forged in a series of n coins. I can determine the minimum and the maximum number of weighings required to locate the forged coin. One weighing is needed to find the forged if it is located in the first pair in a series of coins. The difference occurs when talking about a series with an even or an odd number of coins. The number of weighings is maximum if the forged is located in the last pair of an even series. The number of weighings is maximum if the forged is located in the last three places of an odd series.

Number of coins [n]	2	4	6	8	10	12	...	**n**
Maximum number of weighing	1	2	3	4	5	6		$\frac{\mathbf{n}}{\mathbf{2}}$

Table 2: Number of weighings in relation to number of coins, number of coins in a series is even, $n = 2, 4, 6, 8, ...$

If there is an even number of coins in a series, **1** to maximum $\frac{\mathbf{n}}{\mathbf{2}}$ weighings is needed to locate the forged coin.

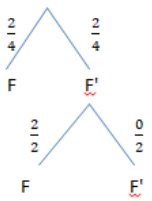
Number of coins [n]	3	5	7	9	11	...	**n**
Maximum number of weighing	1	2	3	4	5		$\frac{\mathbf{n-1}}{\mathbf{2}}$

Table 3: Number of weighings in relation to number of coins, number of coins in a series is odd, $n = 3, 5, 7, 9, ...$

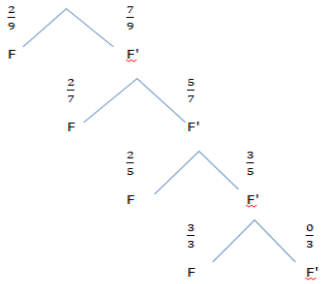
If there is an odd number of coins in a series, **1** to maximum $\frac{\mathbf{n-1}}{\mathbf{2}}$ weighings are needed to locate the forged coin.

Let X be a random variable that represents number of weighings needed to find a forged coin among a series of n coins that are divided into pairs. The probability function can be calculated (Table 1, table 5). The number of weighings is presented by x and number of coins is presented by n.

n = 4



n = 9



1.weighing	$P(X = 1) = \frac{2}{4}$	$P(X = 1) = \frac{2}{9}$
2. weighing	$P(X = 1) = \frac{2}{4} \cdot \frac{2}{2} = \frac{2}{4}$	$P(X = 1) = \frac{7}{9} \cdot \frac{2}{7} = \frac{2}{9}$
3. weighing		$P(X = 1) = \frac{7}{9} \cdot \frac{5}{7} \cdot \frac{2}{5} = \frac{2}{9}$
4. weighing		$P(X = 1) = \frac{7}{9} \cdot \frac{5}{7} \cdot \frac{3}{5} \cdot \frac{3}{3} = \frac{3}{9}$

Table 4: A tree diagram to find values of $P(X = x)$ (F: forged coin found and F': forged coin not found)

If $n = 9$, the coins are arranged into 4 pairs and one remaining coin. The coins in pairs are weighted against each other. The probability of detecting a forged coin after the first weighing is $\frac{2}{9}$, since two coins out of nine, or the first pair, are weighed. Otherwise, the forged coin is among the remaining 7 coins not yet weighed. When weighing the second pair of coins, the forged coin can be found within the first pair with a probability $\frac{2}{7}$. If these two coins are also of the same weight, the forged coin will be found in one of the remaining 5 coins and so on.

The probability distribution for X is presented below. When familiar with a probability distribution for X, we can calculate the average number of weighings needed, when coins are divided into pairs, to find the forged.

x	**n = 2**	**n = 3**	**n = 4**	**n = 5**	**n = 6**	**n = 7**	**n = 8**	**n = 9**	...
	P(X = x)	**P(X = x)**	**P(X = x)**	**P(X = x)**	**P(X = x)**	**P(X = x)**	**P(X = x)**	**P(X = x)**	
1	$\frac{2}{2}$	$\frac{3}{3}$	$\frac{2}{4}$	$\frac{2}{5}$	$\frac{2}{6}$	$\frac{2}{7}$	$\frac{2}{8}$	$\frac{2}{9}$	
2			$\frac{2}{4}$	$\frac{3}{5}$	$\frac{2}{6}$	$\frac{2}{7}$	$\frac{2}{8}$	$\frac{2}{9}$	
3					$\frac{2}{6}$	$\frac{3}{7}$	$\frac{2}{8}$	$\frac{2}{9}$	
4							$\frac{2}{8}$	$\frac{3}{9}$	
...									

Table 5: Probability distribution $P(X = x)$ for X, $2 \leq n$

For a discrete random variable the expected value is calculated by summing the product of the value of the random variable and it's associated probability, taken over all of the values of the random variable. It represents an average value of a variable – average number of weighings that is expected over many trials of the experiment. [4] The formula for calculation of E(X):

$$\mathbf{E(X)} = \sum \mathbf{x} \cdot \mathbf{P(X = x)}.$$

(4)

Number of coins [n]	**Average number of weighings / expected value [E(X)]**
2	$E(X) = 1 \cdot \frac{2}{2} = 1$
3	$E(X) = 1 \cdot \frac{3}{3} = 1$
4	$\mathrm{E(X)} = 1 \cdot \frac{2}{4} + 2 \cdot \frac{2}{4} = 1\frac{1}{2}$
5	$\mathrm{E(X)} = 1 \cdot \frac{2}{5} + 2 \cdot \frac{3}{5} = 1\frac{3}{5}$
6	$\mathrm{E(X)} = 1 \cdot \frac{2}{6} + 2 \cdot \frac{2}{6} + 3 \cdot \frac{2}{6} = 2$
7	$\mathrm{E(X)} = 1 \cdot \frac{2}{7} + 2 \cdot \frac{2}{7} + 3 \cdot \frac{3}{7} = 2\frac{1}{7}$
8	$\mathrm{E(X)} = 1 \cdot \frac{2}{8} + 2 \cdot \frac{2}{8} + 3 \cdot \frac{2}{8} + 4 \cdot \frac{2}{8} = 2\frac{1}{2}$
9	$\mathrm{E(X)} = 1 \cdot \frac{2}{9} + 2 \cdot \frac{2}{9} + 3 \cdot \frac{2}{9} + 4 \cdot \frac{3}{9} = 2\frac{2}{3}$
...	...

Table 6: Average number of weighings in relation to number of coins

Where $\mathrm{n} = 2, 4, 6, 8$ ... the expected value is $\mathrm{E(X)} = 1, 1\frac{1}{2}, 2, 2\frac{1}{2}, \ldots$
Common formula for $\mathrm{E(X)}$:

$$\mathbf{E(X)} = \sum_{x=1}^{\frac{n}{2}} \boldsymbol{x} \cdot \frac{\mathbf{2}}{\boldsymbol{n}}.$$

Where $\mathrm{n} = 3, 5, 7, 9, \ldots$ the expected value is $\mathrm{E(X)} = 1, 1\frac{3}{5}, 2\frac{1}{7}, 2\frac{2}{3}, \ldots$
Common formula for $\mathrm{E(X)}$:

$$\mathbf{E(X)} = \left(\sum_{\mathbf{x=1}}^{\frac{\mathbf{n-3}}{2}} \mathbf{x} \cdot \frac{\mathbf{2}}{\mathbf{n}} \right) + \frac{\mathbf{n-1}}{\mathbf{2}} \cdot \frac{\mathbf{3}}{\mathbf{n}}.$$

The number of weighings when using the 2. method is much smaller than the number of coins being weighted. The number of weighings is much smaller than in the first method.

3. Method: joining coins into three groups and weighing the groups against each other, $\mathbf{n \geq 3}$

There are $\mathrm{n} = 3, 4, 5, \ldots$ coins. One coin is forged, its location in a series of n coins is unknown. There are two possible outcomes, depending on the number of coins in the series. The minimum and maximum number of weighings if $n = 3, 4, 5, 6, 7, 8$ differ from each other (Table 7). If $n = 6$ and the forged coin is the second coin in the series, the scale is not in balance after the first weighing, but leans on the side of the first group, where forged coin is found. Coins from the first group are weighted against each other, the forged is found after two weighings. The latter is true for $4 \leq n \geq 8$. If $n = 3, 9, 27, 81$ the minimum and maxiumum number of weighings are the same (Table 8).

Number of coins [n]	3	4	5	6	7	8	9	10	15	25	27	50	81
The minimum number of weighing	1	1	1	2	1	2	2	2	2	2	3	2	4
The maximum number of weighing	1	2	2	2	2	2	2	3	3	3	3	4	4

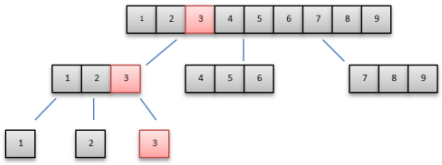
Table 7: Number of weighings in relation to number of coins, applying 3. method

The maximum and the minimum number of weighings are equal when $n = 3, n = 9, n = 27, n = 81$.

Number of coins [n]	$3 = 3^1$	$9 = 3^2$	$27 = 3^3$	...	$\mathbf{n = 3^m}$
Number of weighings [m]	1	2	3		**m**

Table 8: Number of weighings in relation to number of coins

If n presents the number of coins and m presents number of weighings, at least m weighings must be performed to locate the forged coin if $n = 3^m$. 3. method is presented below with a tree diagram. (Picture 1)



Picture 1: Tree diagram for $n = 9$, forged coin is selected randomly, coloured red, $m = 2$. Source: own.

The following is an example for $n = 9$ coins, where the forged coin is the third coin of the series. (Picture 1). A group of nine coins is divided into three groups of three. The first group is weighted against the second group. The scale leans to the side of the first group, where the forged coin is. One weighing is used, coins from 4-9 eliminated. The remaining coins are weighted against each other, the first against the second coin. Since the scale is in balance, the forged coin must be coin number 3. Two weighings were needed to locate the forged coin. The number of weighings does not depend on the location of the forged coin. The number of weighings needed to locate the forged coin is the smallest when using the 3. method.

The question asked when trying to locate the forged coin was: On which side of the scale is the forged coin located and which coin is forged? There are three possible answers:

1. L…left side of the scale is heavier, therefore the forged coin is on the left,
2. R...right side of the scale is heavier, therefore the forged coin is on the right,
3. B…balance, the forged coin is not among all weighted coins.

If the question has three possible answers, then if there are m questions there are 3^m possible answers [2].

In the following part of the article a different question with two possible answers could be posed: Is one group of coins heavier than the other [1]?

1. Yes,
2. No.

There are two possible answers to this question; therefore, there are **2^m possible answers that can answer m questions** [10].

2 TESTING OF BLOOD SAMPLES

When operating with coins the number of weighings needed to locate the forged coin is counted. "Counting numbers (also known as natural numbers) are those which are used to count physical objects in the real world, such as $0, 1, 2, 3, 4, ...$" [10]. When operating with blood samples, however, measurement is used. "Measurement is the procedure or method of identifying the relationship of two numbers" [5].
I will observe the relationship between the expected value of infected samples (there can be more than one) and consequently, the number of tests that need to be performed if there are n infants and infection occurs with probability p.
One of the possibilities of testing is to test each blood sample independently. If there are three infants, three samples of blood are withdrawn and three tests are performed, meaning that to test for infection for n infants n tests are performed. This method would be efficient if there were a lot of infected infants, however, it is long-lasting. I decided to focus on rare abnormalities that affect a small number of infants and use a different method, which would pool infants into groups and in each group, blood samples would be combined together [7].

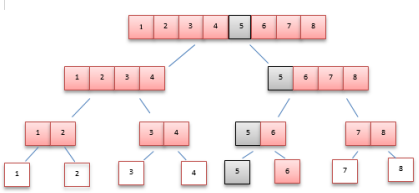
When testing groups of blood samples, there are two possible outcomes [1]

1. P…at least one of blood samples and therefore its donor (infant) in grouped sample shows sign of infection (test is positive),
2. N…all blood samples of donors (infants) are healthy (test is negative).

In a population, a sample of n infants is chosen at random. A rare abnormality occurs in the sample with probability p. Binary protocol is a method that would reduce the number of tests needed to find the infected sample.

3 BINARY PROTOCOL FOR RARE ABNORMALITIES

There are n blood samples. Since it is possible that zero samples are infected because of a rare abnormality a test is performed on a group of n samples. If the test is negative (N), the testing is finished. If the test is positive (P), Binary protocol is used. Group of n samples is split into two smaller groups, tested individually, two tests are performed. If at least one of the groups tests positive, the infected sample is located in that group.
The group with infected sample is tested again using Binary protocol. It is divided into two smaller groups, both are tested and the positive result shows in which group the infected sample is located. The process is repeated until the infected sample is found. Binary protocol can be shown by using a tree diagram.



Picture 2: $n = 8$, one random sample is infected ($X = 1$), coloured grey, dividing by Binary protocol. Source: own.

For $n = 8 = 2^3$ the number of dividing samples into groups is $m = 3$. The number of tests $T = 7$. The number of tests performed is smaller than the number of samples.

Number of samples [**n**]	$4 = 2^2$	$8 = 2^3$	$16 = 2^4$	$32 = 2^5$	...
Number of dividing into groups [**m**]	2	3	4	5	
Number of tests [**T**]	5	7	9	11	

Table 9: The number of tests for $n = 2^m$ when $X = 1$

There are $n = 2^m$ $(m \geq 3)$ blood samples and the number of infected samples is X $(X = 1)$. If the number of samples is bigger than 4, the number of tests is smaller than the number of samples.

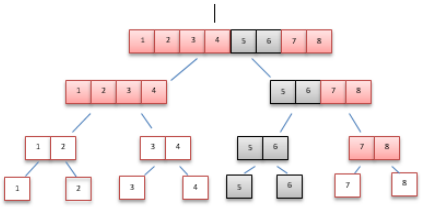
Is there a correlation between the number of tests, number of dividing samples into groups and number of infected samples?

Number of samples [**n**]	$4 = 2^2$	$8 = 2^3$	$16 = 2^4$	$32 = 2^5$	...
Number of tests [**T**]	5 $= 1 + 2 \cdot 1 \cdot 2$	7 $= 1 + 2 \cdot 1 \cdot 3$	$9 = 1 + 2 \cdot 1 \cdot 4$	11 $= 1 + 2 \cdot 1 \cdot 5$	

Table 10: Correlation between number of samples and number of tests for $X = 1$

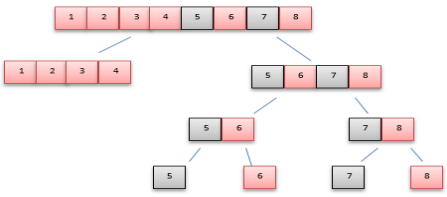
When $X = 1$, it follows: $$\mathbf{T = 1 + 2 \cdot X \cdot m}. \quad (2)$$

Can the formula be applied if there are two or more infected samples? Pictures below present Binary protocol for $n = 8$ and $X = 2$, for randomly located infected samples, coloured grey.



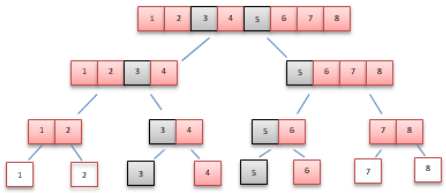
For $n = 8$ is $m = 3$.The number of tests $T_1 = 7$. (Picture 3.1)

Picture 3.1: $n = 8$, $X = 2$, infected samples are in the same group after the first binary division. Source: own.



For $n = 8$ is $m = 3$.The number of tests $T_2 = 9$. (Picture 3.2)

Picture 3.2: $n = 8$, $X = 2$, infected samples are in the same group after the first binary division. Source: own.



For $n = 8$ is $m = 3$. The number of tests $T_3 = 11$.
(Picture 3.3)

Picture 3.3: $n = 8$, $X = 2$, infected samples are in a different group after the first binary division. Source: own.

The number of tests needed to find the infected samples when $n = 8$ and $X = 2$ is the smallest in picture 6.1 and the largest in 6.3, where the number of tests is bigger than the number of samples being tested. Therefore, I assume that the location of the sample after the first binary division determines whether the number of tests will exceed the number of samples.

T_1, T_2 present the number of tests if the infected samples occur in the same group after first binary division.
T_3 presents the number of tests if the infected samples occur in different groups after first binary division.

Number of samples [n]	$4 = 2^2$	$8 = 2^3$	$16 = 2^4$	$32 = 2^5$	...
Number of dividing into groups [m]	2	3	4	5	
Number of tests [T]	$T_1 = T_2 = 5$ $T_3 = 7$	$T_1 = 7$ $T_2 = 9$ $T_3 = 11$	$T_1 = 9$ $T_2 = 13$ $T_3 = 15$	$T_1 = 11$ $T_2 = 17$ $T_3 = 19$	

Table 11: The number of tests for $n = 2^m$ when $X = 2$

There are $n = 2^m, (m = 3, m = 4, \dots)$ blood samples, the number of infected samples is X, $(X = 2)$. I use the formula $\mathbf{T = 1 + 2 \cdot X \cdot m}$ when there are two infected samples.

Number of samples [**n**]	$4 = 2^2$	$8 = 2^3$	$16 = 2^4$	$32 = 2^5$	...
Number of tests [**T**]	$1 + 2 \cdot 2 \cdot 2 = 9$	$1 + 2 \cdot 2 \cdot 3 = 13$	$1 + 2 \cdot 2 \cdot 4 = 17$	$1 + 2 \cdot 2 \cdot 5 = 21$	

Table 12: Number of tests in relation to number of samples for $X = 2$ if formula $\mathbf{T = 1 + 2 \cdot X \cdot m}$ is used

The number of tests calculated by formula $\mathbf{T = 1 + 2 \cdot X \cdot m}$ is larger than the ones counted in a tree diagram. The differences occur because when using the formula we cannot predict where in a series of n samples the infected blood sample is located. When counting tests in a tree diagram we can predict the position of infected samples. The largest number of tests performed can be calculated by formula $\mathbf{T \leq 1 + 2 \cdot X \cdot m}$.

Proof: If we have 2^m blood samples, there are $(m + 1)$ groups in the tree diagram. The uppermost group, which is tested first and m groups below it. Each group below the first is tested only in case if one group above shows signs of infection. The collective number of tests performed on each level is $2 \cdot X$ so it is two times bigger than the number of infected samples. Because there are m levels, it follows [2]
$2 \cdot m \cdot X$.

The largest number of tests performed is $T \leq 1 + 2 \cdot m \cdot X$.

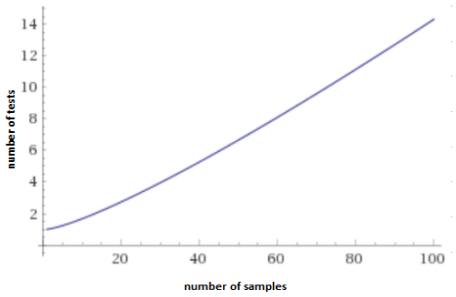
The exact number of infected infants is often unknown. The infection occurs among n samples, with probability p. When we are familiar with the probability, we can calculate the expected value of infected samples:

$$\mathbf{E(X) = n \cdot p}. \quad (4)$$

From formula $T \leq 1 + 2 \cdot X \cdot m$, a new evaluation of number of tests is needed to find the infected sample.
If $E(X) = n \cdot p$
and if $n = 2^m$ then $m = \log_2 n$.
(4)
It follows that

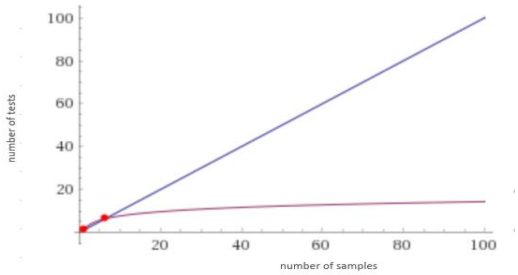
$$\mathbf{T \leq 1 + 2 \cdot n \cdot p \cdot \log_2 n}\ . \quad (2)$$

Graph of function (blue line)



$f(n) = 1 + 2 \cdot n \cdot p \cdot \log_2 n$
shows that the number of tests is much smaller than the number of samples if the probability $p = 0{,}01$.

Graph 1: Number of tests in relation to number of samples, $p = 0{,}01$, number of samples from $n = 1$ to $n = 100$



Graph of function (blue line)
$f(n) = n$
represents testing each sample separately.

Graph of function (red line)
$f(n) = 1 + 2 \cdot \log_2 n$
represents testing when applying Binary protocol.

Graph 2: Number of tests in relation to number of samples, $p = \frac{1}{n}$, $n = 1$ to $n = 100$

If $n \leq 6$ than the number of tests needed to find the infected sample is larger than the number of samples, which can also be seen in the graph, presented by the line between two red points. If $n \geq 7$ than the number of tests performed to find the infected sample is smaller than the number of all samples. The larger n is, the smaller p is and the number of tests needed to find the infected sample is much smaller than the number of samples.

4 THE EXPECTED VALUE OF TESTS

If there are $n = 2^m$ samples and abnormality occurs with probability p, the normal samples occur with probability
$(1 - p)$, $0 < k \leq m - 1$. [1]

Then the expected value of tests when using Binary protocol is

$$\mathbf{E(n) = 1 + 2 \cdot \sum_{k=0}^{m-1} 2^k \left(1 - (1 - p)^{2^{m-k}}\right)}. \quad (1)$$

Number of samples $[\mathbf{n}]$	**Expected value of tests $\mathbf{E(n)}$**
$4 = 2^2$	$E(4) = 1 + 2 \cdot \sum_{k=0}^{1} 2^k \left(1 - (1 - 0{,}01)^{2^{2-k}}\right) \approx 1 + 2 \cdot 0{,}079 = 1{,}158$
$8 = 2^3$	$E(8) = 1 + 2 \cdot \sum_{k=0}^{2} 2^k \left(1 - (1 - 0{,}01)^{2^{3-k}}\right) \approx 1 + 2 \cdot 0{,}236 = 1{,}472$
$16 = 2^4$	$E(16) = 1 + 2 \cdot \sum_{k=0}^{3} 2^k \left(1 - (1 - 0{,}01)^{2^{4-k}}\right) \approx 1 + 2 \cdot 0{,}619 = 2{,}238$
$32 = 2^5$	$E(32) = 1 + 2 \cdot \sum_{k=0}^{4} 2^k \left(1 - (1 - 0{,}01)^{2^{5-k}}\right) \approx 1 + 2 \cdot 1{,}835 = 4{,}670$
$64 = 2^6$	$E(64) = 1 + 2 \cdot \sum_{k=0}^{5} 2^k \left(1 - (1 - 0{,}01)^{2^{6-k}}\right) \approx 1 + 2 \cdot 3{,}504 = 8{,}008$
$128 = 2^7$	$E(128) = 1 + 2 \cdot \sum_{k=0}^{6} 2^k \left(1 - (1 - 0{,}01)^{2^{7-k}}\right) \approx 1 + 2 \cdot 7{,}732 = 16{,}464$
$256 = 2^8$	$E(256) = 1 + 2 \cdot \sum_{k=0}^{7} 2^k \left(1 - (1 - 0{,}01)^{2^{8-k}}\right) \approx 1 + 2 \cdot 16{,}387 = 33{,}774$

Table 13: Expected value of tests when $p = 0{,}01$

From Table 13 we can see that expected value of tests $E(n)$ is much smaller than the number of samples n. The difference between n and $E(n)$ is greater when the number of samples is greater.

5 CONCLUSION

Based on upper calculations, pictures and graphs I have reached the aim of my assignment. It is possible to see that Binary protocol is useful if we have a large number of blood samples because it reduces the number of tests when looking for infected samples.

However, when the number of blood samples is small, the number of tests performed using Binary protocol is larger than the number of samples. This could be seen as a disadvantage of the method, which is not as useful when dealing with a small number of blood samples. Moreover, there are deviations between the calculated value of tests and the number of tests that were counted in a tree diagram. In the latter, I can predict the location of an infected sample.

Although it may be seen as merely theoretical exploration today, blood testing with Binary protocol was used in real life. This method was practised during the Second World War to test men for syphilis. Because individual testing was too expensive, groups of blood samples were made and Binary protocol was used to detect the disease [6]. There are several reasons why the method is not (often) used today. Firstly, the medical sciences have advanced and discovered new ways to deal with abnormalities of infants. For example, in Australia, a new non-invasive maternal blood test for detecting fetal abnormalities was discovered. It can detect certain chromosomal disorders [8]. Secondly, there are ethical hesitations. If each infant is tested separately only a small sample of blood is needed. However, when testing with Binary protocol a larger sample of infant's blood is required for the same result, which is on the other hand, achieved in a shorter time and with less money. Thirdly there are problems connected to blood mixing such as "lack of homogeneity of a blood sample can lead to incorrect results" [9]. And lastly, I have used Binary protocol to test for rare abnormalities. The latter would occur very rarely or would often not even occur. However, in the world we are often dealing with abnormalities that occur with a higher probability.

Bibliography

[1] Ball, K. (2004). *Pools of blood*. Retrieved: 17. 10. 2018: https://plus.maths.org/content/pools-blood

[2] Ball, K. M. (2003). *Strange Curves, Counting Rabbits, and Other Mathematical Explorations.* Princeton: Princeton University Press, 130-139.

[3] Blagostinčar Blagotinšek Ana, R. N. (n.d.). *Guganje in tehtanje = wobble and weighing.* Retrieved: 2. 12. 2018: http://fibonacci-project.si/gradiva/fibo_gradiva/04guganje_tehtanje.pdf

[4] Buchanan Laurie, F. J. (2012). *Mathematics Standard Level.* Oxford: Oxford University Press, 115, 284, 523-524, 535.

[5] Dictionary, M. (n.d.). *What is measurement - Definition and Meaning*. Retrieved: 11. 12. 2018: https://www.easycalculation.com/maths-dictionary/measurement.html

[6] Dorfman, R. (1943). *The Detection of Defective Members of Large Populations*. Retrieved 10. 12. 2018: https://projecteuclid.org/download/pdf_1/euclid.aoms/1177731363

[7] Hwang, F. K. (1972). *A Method for Detecting all Defective Members in a Population by Group Testing*. Retrieved from JSTOR: https://www.jstor.org/stable/2284447?seq=1#page_scan_tab_contents

[8] Lab Tests Online. (2013). *New blood test for detecting fetal abnormalities available in Australia*. Retrieved: 24. 12. 2018: https://www.labtestsonline.org.au/news/pathology-update/new-blood-test-for-detecting-fetal-abnormalities-a

[9] Narayanan, S. (2005). *Preanalytical issues related to blood sample mixing*. Retrieved: 23. 12. 2018: https://acutecaretesting.org/en/articles/preanalytical-issues-related-to-blood-sample-mixing

[10] StudyPad, Inc. (n.d.). *Common Core Math Vocabulary*. Retrieved: 11. 12. 2018: https://www.splashmath.com/math-vocabulary/counting-and-comparison/count

VABILO AVTORJEM

Dianoia (grško διάνοια) po Platonu označuje védenje, razmišljanje o modelih stvarnosti, o naravoslovno-matematičnih in tehničnih temah. Uporabljajo ga matematiki (modeliranje) in znanstveniki (formuliranje problema), inženirji (načrtovanje sistema). Opredeljuje kompetenco, proces ali rezultat diskurzivnega razmišljanja, za razliko od neposrednega razumevanja obravnavane tematike. Aristotel to védenje naprej razdeli na teoretično (episteme) in praktično (phronesis).

Dianoia po Platonu torej označuje vmesni nivo človeškega spoznanja, prehod od intuitivnih občutkov do najglobljega spoznanja dejanskosti. Tako je idealna oznaka za objave v pričujoči reviji, ki povezujejo teoretična, znanstvena izhodišča z njihovo uporabno namembnostjo. Študentje, avtorji teh člankov, ste na prehodu od učenja k delu, od teoretičnega h konkretnemu, ki vas bo pripeljalo do kruha, do dela, s katerim boste odigrali svojo vlogo v družbi. Na tem prehodu pa poleg znanja, ki ga ponuja redno izobraževanje, potrebujete tudi izkušnje s konkretnih izzivov in mehke kompetence sodelovanja v ekipah delodajalcev, k čemur vas spodbuja in vam pri tem pomaga revija Dianoia.

V reviji bomo objavljali poljudne in strokovne članke s področja naravoslovja, matematike ali znanosti, ki uporabljajo znanja teh področij. Ciljna publika bralcev so v prvi vrsti delodajalci, ki tovrstna znanja potrebujejo in želijo izvedeti, kaj je kdo zanimivega razmislil na njihovem področju. V drugi vrsti so ciljna publika študentje, ki iščejo zamisli za svojo poklicno pot in lahko v reviji najdejo navdih za lastna raziskovanja in iskanje stikov s trgom dela.

Za kakovost izdelkov bo skrbel uredniški odbor in uredniški svet, v katerih so vrhunski strokovnjaki, povezani s področji, ki jih revija obravnava. Članki bodo anonimno recenzirani, o objavi pa na podlagi recenzije odloča uredniški odbor. Priporočljivo je, da avtorji besedilo spremenijo v skladu s priporočili recenzentov in da popravljeni članek z utemeljitvijo sprejema ali zavrnitve sprememb ponovno pošljejo v pregled. Uredništvo lahko objavo članka zavrne, če vsebinsko ali po merilih kakovosti ne ustreza standardom revije, o čemer avtorje obvestimo v najkrajšem možnem času.

S prispevkom v reviji bodo avtorji spodbujali širjenje znanja s področja naravoslovja in matematike ter tehnike oziroma izobraževanja teh področij in svoje poglede prenašali na trg dela in na prihajajoče generacije.

NAVODILA AVTORJEM

Avtorje prosimo, da pri pripravi članka upoštevajo naslednja navodila.

Če je članek napisan v slovenščini, naj ima angleški prevod naslova, povzetka in ključnih besed. Veseli bomo tudi prispevkov v angleščini, ki pa morajo imeti naslov, razširjen povzetek v obsegu 300 – 400 besed in ključne besede v slovenščini. Ključnih besed naj bo do šest.

Prispevki naj bodo zanimivi za širši krog bralcev. Ključna je intuitivna predstavitev zamisli in rezultatov, podrobnosti pa lahko ostanejo prihranjene za morebitni znanstveni članek, ki bi bil nadgradnja članka, objavljenega v reviji Dianoia.

Članek naj vsebuje naslov, ime avtorja (avtorjev) in sedež ustanove, kjer avtor(ji) dela(jo). Sledi naj povzetek, z največ 150 besedami, seznam ključnih besed in besedilo, ki ne presega 3000 besed. Besedilo naj bo zapisano v urejevalniku besedil MS Word 2010 oz. kasnejši ali LaTeX in naj uporablja objavljeno predlogo. Slike in tabele morajo biti oštevilčene in imeti natančen opis, da jih lahko razumemo brez preostalega besedila. Slike v elektronski obliki naj bodo visoke kakovosti v formatu PNG ali JPEG.

Prispevek v PDF obliki pošljite na naslov dianoia@um.si z zadevo: »Za revijo Dianoia«. Če bo sprejet v objavo, vas bomo prosili za izvorno obliko prispevka.