

Řešení a generování deterministických a nedeterministických logických úloh.

Solving and generation of deterministic and
nondeterministic logical tasks.

Bc. Pavel Vyhlídal



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2012/2013

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Pavel Vyhlídal**
Osobní číslo: **A11514**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **kombinovaná**

Téma práce: **Řešení a generování deterministických
a nedeterministických logických úloh**

Zásady pro vypracování:

1. Seznamte se s různými logickými úlohami a problematikou jejich řešení.
2. Prostudujte možnosti generování logických úloh pomocí deterministických a nedeterministických algoritmů, projděte některá stávající řešení.
3. Vyberte vhodné řešitelné úlohy a navrhnete algoritmy umožňující generování a kontrolu správnosti zadání logických úloh.
4. Implementujte tyto algoritmy v jazyce PHP a zpřístupněte přes webové rozhraní. Umožněte, aby generované zadání některých úloh bylo jak textové v přirozeném jazyce, tak grafické (kulturně indeferentní).
5. Věnujte pozornost zabezpečení webové aplikace.
6. Vytvořte funkci logování pro shromažďování odezvy uživatelů při řešení těchto úloh tak, aby bylo možno vyhodnotit složitost a problémy při řešení jednotlivých úloh a jejich variant.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. ABELSON, H., SUSSMAN G. J. Structure and Interpretation of Computer Programs, MIT Press, 1984. ISBN 0-262-01077-1.
2. STANGROOM, J. Einstein's Riddle: Riddles, Paradoxes, and Conundrums to Stretch Your Mind, Elwin Street Productions, 2009. ISBN 978-1-59691-665-4.
3. MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J. a kolektiv Umělá inteligence (3), Nakladatelství Academia, 2001. ISBN 978-80-200-0472-6.
4. MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J. a kolektiv Umělá inteligence (5), Nakladatelství Academia, 2007. ISBN 978-80-200-1470-2.
5. RUSSELL, S., NORVIG, P. Artificial Intelligence: A Modern Approach, 3rd Edition, Prentice Hall, 2009. ISBN 978-0-13-604259-4 .
6. ZELINKA, I. Umělá. inteligence – hrozba nebo naděje. BEN – technická literatura, 2003. ISBN 80-7300-068-7.
7. GARDNER, M. The Colossal Book of Mathematics. Norton, 2001. ISBN 978-0-393-02023-6.

Vedoucí diplomové práce: **Ing. Roman Šenkeřík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **22. února 2013**

Termín odevzdání diplomové práce: **22. května 2013**

Ve Zlíně dne 22. února 2013



prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem diplomové práce je generování logických úloh pomocí deterministických a nedeterministických algoritmů. V práci jsou vybrány vhodné řešitelné úlohy a následně jsou k nim navrženy algoritmy umožňující generování a kontrolu správnosti zadání logických úloh. Tyto algoritmy jsou implementovány v jazyce PHP a zpřístupněny přes webové rozhraní. Odezvy uživatelů při řešení těchto úloh jsou zaznamenávány do logů pro další zpracování. V závěru je vyhodnocena složitost a problémy při řešení jednotlivých úloh a jejich variant.

Klíčová slova: logické úlohy, zebra puzzle, rekurze.

ABSTRACT

The aim of this study is to generate a logical task using deterministic and non-deterministic algorithms. The work included the selection of appropriate solvable problems as well as the subsequent design of algorithms, which enable generating and checking the correctness of input logic tasks. These algorithms are implemented in PHP language and made available through a web interface. Using logging, feedback from users who dealt with these problems is gathered. Finally, the complexity of the individual problems and their variants, as well as problem which occurred while solving them, are evaluated.

Keywords: brain teaser, zebra puzzle, recursion.

Existuje také definice - hypotéza. Rozum je složitý instinkt, který se ještě nestačil zformovat. Tady se vychází z toho, že instinktivní činnost je vždy účelná, přirozená.

prof. Valentin Pillman

(Arkadij a Boris Stugatští, „Piknik u cesty“)

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD.....	8
I. TEORETICKÁ ČÁST	9
1 INTELIGENCE.....	10
1.1 CO JE INTELIGENCE.....	10
1.2 TEORIE VÍCENÁSOBNÉ INTELIGENCE.....	11
1.3 LOGICKO-MATEMATICKÁ INTELIGENCE.....	12
1.4 VYMEZENÍ.....	13
2 PŘÍKLADY LOGICKÝCH ÚLOH.....	14
2.1 SET	14
2.2 KOSTKA	14
2.3 ZEBRA PUZZLE.....	16
2.4 FRAKTÁLOVY JONATHAN	19
2.5 JAK SI HRAJÍ LIDÉ	20
II. PRAKTICKÁ ČÁST.....	23
3 VYTVOŘENÉ ÚLOHY.....	24
3.1 SET	24
3.2 KOSTKA	27
3.3 ZEBRA PUZZLE.....	29
3.4 VÁHY	37
3.5 INVESTOR.....	38
3.6 SUDRA.....	40
4 PRAKTICKÉ PROVEDENÍ.....	42
4.1 PLATFORMA	42
4.2 MOŽNOSTI ZPRACOVÁNÍ	42
4.3 ZABEZPEČENÍ.....	43
4.4 BODOVÁNÍ	43
4.5 DALŠÍ NÁMĚTY	44
4.5.1 TRIPLIKÁTOR.....	44
4.5.2 DEADLOCK.....	44
ZÁVĚR	46
CONCLUSION	47
LITERATURA A INFORMAČNÍ ZDROJE	48
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	50
SEZNAM OBRÁZKŮ	51
SEZNAM TABULEK.....	52
SEZNAM PŘÍLOH.....	53

ÚVOD

Inteligenci můžeme chápat jako rozumovou schopnost (v tomto případě lidí) řešit nové úkoly nebo neznámé situace, učit se z vlastních nebo cizích zkušeností nebo adaptovat se na nové prostředí. Intelligence je podmíněna geneticky, ale lze ji rozvíjet získáváním nových zkušeností a procvičováním modelových situací. Klíčová je schopnost správného určení podstatných souvislostí a vztahů, podle nichž se v nových situacích orientujeme a nové problémy řešíme. Právě tvorbou logických úloh pro získávání nových zkušeností a procvičování modelových situací se zabývá tato práce.

Cílem této práce je vytvořit aplikace, které jsou schopny generovat logicko-matematické úlohy určené k procvičování případně jako součást testů, například v soutěži Logická olympiáda. K oběma účelům jsou uzpůsobeny graficky.

V práci jsou vybrány úlohy interaktivní, které probíhají v reálném čase komunikací – člověk – počítač a také úlohy jednorázové, po jejichž vygenerování je očekávána odpověď a vyhodnocena jako správná či špatná. Práce je motivována několika existujícími portály s logickými úlohami, z nichž některé motivy byly přejaty, rozšířeny, lokalizovány nebo přizpůsobeny účelu použití. Jsou ale také vytvořeny nové úlohy, které jsou dosud k dispozici jen v tištěné podobě či jejich publikace není autorovi vůbec známa.

Tato práce se zabývá i úlohami, které spadají do kategorie ekonomických her. V této části budou popsány principy, navrženo řešení a vytvořen program, který umožňuje interaktivní hru člověk – počítač, ačkoli tyto hry bývají obvykle hrány interaktivně člověk – skupina. Cílem poté nebude kontrolovat správné či špatné odpovědi, ale sledovat chování lidí v interakci se skupinou.

Dále je popsáno praktické řešení generátorů, které spolu s příloženými skripty může sloužit jako podklad pro jejich úpravy či rozšíření o nové funkce nebo jako zdroj nových úloh.

I. TEORETICKÁ ČÁST

1 INTELLIGENCE

Lidé se liší svými schopnostmi počínat si správně v různých situacích a svými schopnostmi vykonávat některé práce či řešit některé úkoly. Pojem intelligence ve vztahu k rozumové činnosti, který poprvé používal F. Galton (1822-1911), nahradil výraz *habilité* (anglicky *ability*) tedy schopnost, dovednost či chytrost, který zavedl průkopník psychologických testů A. Binet (1857-1911) [1]. Na otázku co je intelligence a jak ji definovat, není dosud jednoznačná odpověď.

1.1 CO JE INTELLIGENCE

Motto: Inteligentní člověk dokáže rozpoznat v chaosu řád a naopak v řádu chaos.

Jak definovat inteligenci?

Definovat tak složitou vlastnost, kterou navíc ještě zcela nerozumíme, je obtížné. Každá definice je omezením, a proto zvláště v tomto případě nemůže být nikdy dokonalá. Různí filozofové či psychologové definovali inteligenci různě, např. takto:

W. Stern: Intelligence je všeobecná schopnost individua vědomě orientovat vlastní myšlení na nové požadavky, je to všeobecná duchovní schopnost přizpůsobit se novým životním úkolům a podmínkám.

D. Wechsler: Intelligence je vnitřně členitá a zároveň globální schopnost individua účelně jednat, rozumně myslet a efektivně se vyrovnávat se svým okolím.

J. P. Guilford: Intelligence je schopnost zpracovávat informace. Informacemi je třeba chápat všechny dojmy, které člověk vnímá.

H. E. Gardner definuje inteligenci jako "biopsychologický potenciál ke zpracování informací, které mohou být využity v kulturním prostředí k řešení problémů nebo vytváření kulturních hodnot" [2]. Podle něj existuje více způsobů, jak řešit problémy, než jen s využitím logické a jazykové intelligence.

Dále jsou podle Gardnera IQ testy zaměřeny především na logickou a jazykovou inteligenci. Při tom úspěšnost při testech dává šanci přijetí na vysokou školu. [3]. Naproti tomu podle Heldinga [4], jsou „standardní IQ testy měří znalosti získané v určitém okamžiku, takže mohou poskytnout pouze zamrzlý pohled na elementární znalosti“. Podle něho nelze posoudit ani předpovědět schopnosti člověka učit se, vstřebávat nové informace či řešit nové problémy.

1.2 TEORIE VÍCENÁSOBNÉ INTELIGENCE

Můžeme rozlišovat více druhů inteligence, někdy taky nazývané dimenze inteligence nebo vícenásobná inteligence člověka (Multiple Intelligences). Každá z těchto inteligencí je schopná řešit určité druhy problémů. [1]

Gardner rozdělil inteligenci do devíti skupin (překlady se mohou lišit):

- Logicko-matematická (Logical-mathematical)
- Vizuálně-prostorová (Spatial)
- Mezilidská (Interpersonal)
- Jazyková (Linguistic)
- Tělesně pohybová (Bodily-kinesthetic)
- Hudební (Musical)
- Vnitřní, osobní (Intrapersonal)
- Přírodopisná (Naturalistic)
- Existenciální (Existential)

Logicko-matematická inteligence nám pomáhá chápat příčinu a důsledek. Díky ní můžeme počítat, zvažovat, provádět matematické a logické operace. Pomáhá při induktivním i deduktivním myšlenkovém procesu. Jedním ze znaků matematicko-logické inteligence je schopnost vnímat pravidelnost mezi vztahy a jevy. Matematicko-logická inteligence je pro Vás převládající, když:

- Dovedu snadno ve své mysli dělat výpočty.
- Matematika, případně i jiné přírodní vědy patřily ve škole k mým oblíbeným předmětům.
- Rád hraji hry nebo řeším hlavolamy, které vyžadují logické myšlení.
- Rád dělám malé „co kdyby“ experimenty (např. „Co by se stalo, kdybych pravidelně zaléval svůj růžový keř dvojnásobným množstvím vody?“).
- Moje mysl hledá struktury, pravidelnosti nebo logické souvislosti ve věcech.
- Zajímám se o nové objevy vědy.
- Jsem přesvědčen, že téměř vše je možno racionálně vysvětlit.
- Někdy myslím v jasných, abstraktních, bezeslovních a neobrazných pojmech.
- Rád hledám logické mezery ve věcech, které lidé říkají a dělají doma i v práci.

- Cítím se spokojenější, kdy se mi něco podařilo změřit, zařadit, analyzovat nebo nějak kvantifikovat.

Lidé, kteří vynikají v prostorové inteligenci, inklinují k myšlení v obrazech [5]. Myslí ve třech dimenzích a dokážou mentálně manipulovat s objekty. Mají silnou představivost.

Prostorově inteligentní lidé často:

- Často vidím jasné vizuální obrazy, když zavřu oči.
- Mám cit pro barvy.
- Běžně používám fotoaparát nebo videokameru na zaznamenání toho, co kolem sebe vidím.
- Těší mne, když mohu sestavovat skládačky, řešit bludiště nebo jiné vizuální hádanky.
- Mám v noci živé sny.
- Mohu běžně najít správnou cestu v neznámém terénu.
- Rád maluji nebo čmárám.
- Geometrie byla pro mne ve škole lehčí než aritmetika.
- Dovedu si snadno představit, jak něco může vypadat, kdyby se na to někdo díval přímo seshora z ptačí perspektivy.
- Raději si prohlížím texty, které jsou hodně ilustrované.

V této práci se budu věnovat především logicko-matematické a prostorové, nicméně některé úlohy jsou směřovány do inteligence interpersonální.

1.3 LOGICKO-MATEMATICKÁ INTELIGENCE

Logicko-matematická inteligence je schopnost, která se uplatňuje ve vědeckém myšlení. Spolu s řečovou schopností (verbální usuzování) je obvykle měřena testy IQ. Vývoj tohoto druhu inteligence zkoumal například Jean Piaget a další psychologové. Výzkumy mozku ukazují, že některé oblasti mozku mají při matematických výpočtech významnější úlohu než jiné. Vlastní mechanismus, který způsobuje, že někteří lidé v matematice mimořádně vynikají, dosud neznáme. Známe však typy činností, které tento druh inteligence posilují [1]. Patří k nim:

- vedení osobních účtů
- plánování výletů a cest

- procvičování počítání z paměti
- výpočet šancí a pravděpodobnosti různých jevů
- odhadování množství
- řešení logických hádanek a hlavolamů

Právě poslední bod je motivací pro tvorbu těchto matematicko-logických úloh.

1.4 VYMEZENÍ

Práce se zabývá úlohami několika typů, přičemž každý typ úlohy má svůj cíl, ať už v trénování mentálních schopností nebo v poznání.

Prvním typem jsou úlohy, které lze řešit zcela deterministicky, souborem kroků, přičemž v každém je přesně určen krok následující. Cílem takovýchto úloh je naučit se analyzovat problém, optimalizovat postup řešení, nalézat v úloze klíčové prvky a odfiltrovat nepodstatné.

Další skupinou úloh jsou úlohy, v nichž se jasně určenými kroky dostanete jen do určité fáze řešení, poté lze postupovat několika cestami, přičemž jenom jedna vede ke správnému řešení. Cílem těchto úloh je vytváření myšlenkových paralelních map, v nichž se učíme pohybovat k cíli. Stejně jako u první skupiny jsou zaměřeny na rozvoj inteligence matematicko-logické a prostorové.

Poslední jsou úlohy, které vůbec nemají racionální řešení. Řeší se ve skupině, přičemž úlohou je maximalizace zisku. Zde je však v rozporu maximalizace zisku jednotlivce a maximalizace zisku skupiny, přičemž zisk jednotlivce je závislý od zisku skupiny. Tyto úlohy nemají optimální řešení, nelze tedy ani určit správný výsledek. Jejich řešení je možné pouze iracionálním chováním členů a proto bych je radil v teorii vícenásobné inteligence do skupiny interpersonální inteligence, sociálních vztahů. Smyslem tvorby těchto úloh nebude rozvoj logické-matematické, sociální či jiné inteligence, ale poznání, výzkum chování jednotlivce, který tyto úlohy řeší.

2 PŘÍKLADY LOGICKÝCH ÚLOH

V této části je popsáno několik existujících úloh, portálů či her, z nichž některé byly inspirací pro praktickou část.

2.1 SET

Je karetní hra, hraná v reálném čase. Vytvořila ji Marsha Falco [5] v roce 1974. Obsahuje 81 karet. Na každé kartě je obrázek se čtyřmi atributy, každý s jednou ze třech hodnot:

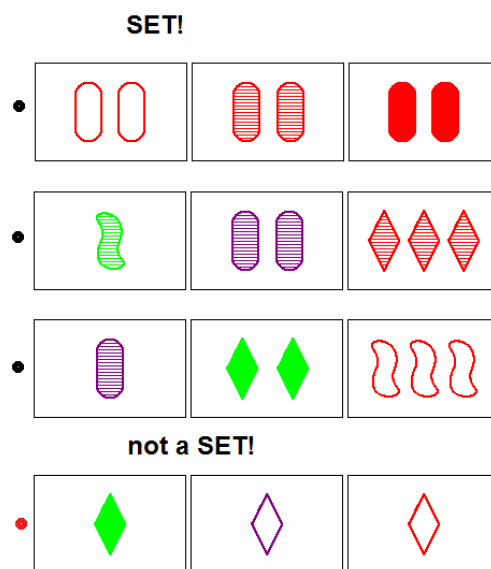
- barva – zelená, fialová, červená
- počet – 1, 2, 3
- tvar – ovál, diamond a squiggle (angl., česky kosočtverec a vlnka)
- výplň – prázdný, pruhovaný, plný

Celkový počet těchto karet je dán vzorcem 3^4 , v tomto případě 3^4 , tedy 81 karet. Principem hry je najít 3 karty (Obrázek 1), na které má každý atribut buď vždy stejné nebo rozdílné hodnoty. Z analýzy [6] vyplývá, že ve hře je možných 1080 různých setů, na stole může být vyloženo asi $7 \cdot 1013$ kombinací karet (při 12 kartách na stole) a také že průměrný počet SETů při 12 kartách je 2,78. Nicméně se také můžou vyskytnout situace, že na stole není žádný SET, a poté je třeba přidat (3) karty. Maximálně je možná sestava 20 karet, na kterých není platný SET. Díky tomu také hra nemá pevně daný počet odehraných kol.

Tato hra je dostupná i pro on-line hry, kdy hraje hráč sám nebo proti „robotovi“, s různým počtem kol či různým bodováním.

2.2 KOSTKA

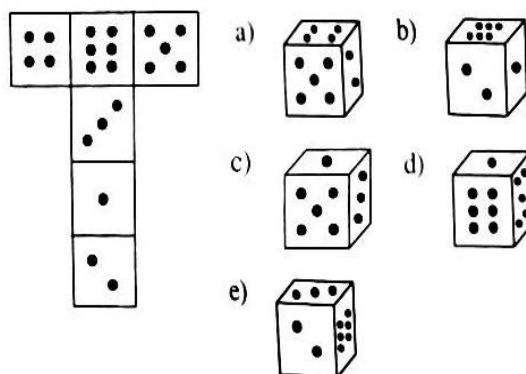
Cílem úloh zobrazení na kostce je obvykle rozvoj prostorové představivosti. Základem je kostka (můžou být i jiné tvary), na jejíchž stranách jsou různé symboly či texty.



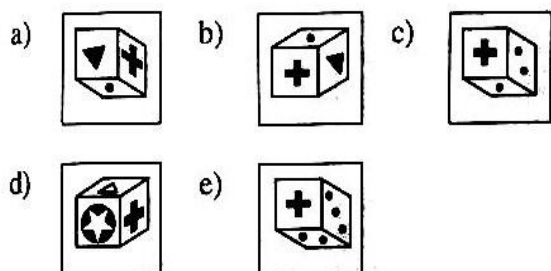
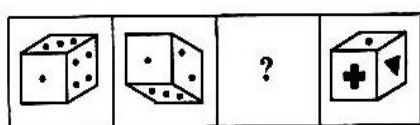
Obrázek 1: SET - příklady správné / nesprávné sestavy.

Kostka může být zobrazena v rozvinutém stavu, nebo v 3D pohledu, který zobrazuje 3 strany. Vyobrazena může být klasická hrací kostka (Obrázek 2) nebo kostka se symboly (Obrázek 3) [7].

Úkolem může být identifikovat 3D pohled s rozvinutým plánem, identifikovat 3D pohledy mezi sebou nebo určovat logicky



Obrázek 2: Kostky - příklad úlohy na kostce [7].

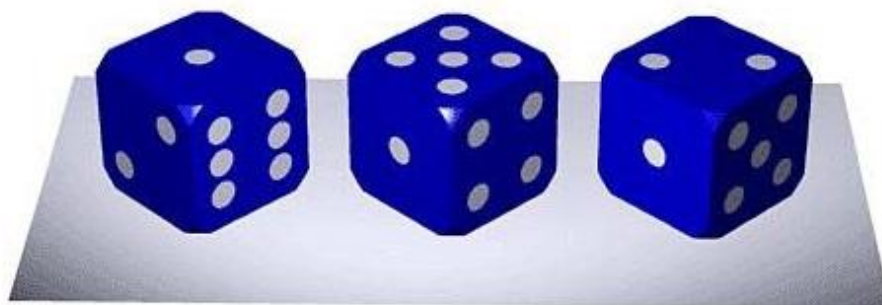


Obrázek 3: Kostky - příklad úlohy na kostce [7].

obsah stran, např. podle počtu teček.

Jedná se o tři shodné kostky v různých pohledech (Obrázek 4). Jejich výrobce neznal pravidlo o součtu protilehlých stran, a tak vytvořil kostku dle svých představ.

Na kterých číslech jednotlivé kostky stojí? [8]



Obrázek 4: Příklad: Na kterých číslech jednotlivé kostky stojí? [8]

2.3 ZEBRA PUZZLE

Dalším typem jsou „LOGIC PUZZLE“, což jsou hádanky odvozeným z matematické oblasti dedukce. Základem jsou indicie (premisy, tipy), z nichž je možné vydedukovat obsah jednotlivých členů mřížky.

Tyto úlohy se vyskytují v mnoha formách, například jako „Dinesman's multiple-dwelling problem“ na [9] Rosetta Code:

„Baker, Cooper, Fletcher, Miller, and Smith live on different floors of an apartment house that contains only five floors. Baker does not live on the top floor. Cooper does not live on the bottom floor. Fletcher does not live on either the top or the bottom floor. Miller lives on a higher floor than does Cooper. Smith does not live on a floor adjacent to Fletcher's. Fletcher does not live on a floor adjacent to Cooper's. Where does everyone live?“.

Andrea, Broňa, Cecílie, Dana a Eva bydlí v pětipatrovém domě v různých patrech. Andrea nebydlí úplně nahoře. Broňa nebydlí v přízemí. Cecílie nebydlí ani pod střechou ani v přízemí. Dana bydlí nad Broňou. Eva nebydlí přímo nad ani pod Cecílkou. A Cecilka nebydlí přímo pod ani nad Broňou.

Jedno velmi propracované zadání takové úlohy uvedl na svých stránkách Ing. Zbyněk Křivka, Ph.D. (FIT VUT Brno) [10], pravděpodobně překlad z německého zadání:

PĚTIPOSCHOĐOVÝ DŮM

Zadání: Zjisti, kdo bydlí ve kterém patře, kolik má dětí, kolik platí nájem a jaké má zaměstnání.

- Ve výškovém domě je 5 bytů. (přízemí a 1., 2., 3., 4. poschodí)
- Rodina Müllerových má jako jediná 4 děti.
- V prostředním bytě bydlí správce.
- Paní Meierová platí za svůj byt o 60 korun méně než lékař ve 3. poschodí.
- Rodina v nejvyšším patře má 5 dětí.
- Rodina Kernů má o 1 dítě méně než učitelka.
- Pan Kaufmann je zástupce jedné firmy.
- Učitelka platí za svůj byt 1740 korun.
- V nejvyšším patře bydlí domovník.
- Lékař platí za byt 1800 korun.
- V bytě, který stojí 1760 korun, nebydlí žádné děti.
- Ve 2. poschodí mají dvakrát tolik dětí jako ve 3. poschodí.
- Pan Hanz bydlí přímo nad rodinou Kernů.

- V bytě pod domovníkem žijí 2 děti.
- Müllerovi platí za byt 1770 korun.
- Střešní byt je o 10 korun lacinší než přízemí.
- Zástupce platí o 30 korun víc než domovník.

V tomto zadání nejsou hodnoty atributů zadány jen výčtem, ale také diferenciálně vůči ostatním (položka 1730 není v zadání vůbec uvedena). Také je potřeba zadání rozdělit na elementární vztahy, protože v některých řádcích se vyskytují 2 vztahy najednou.

Velmi propracovaný generátor takových to úloh je v provozu na stránkách www.mensus.net [11]. Jejich program vygeneruje zadání (Obrázek 5) s možností vyplnění odpovědi v grafické formě:

K zadání jsou vygenerovány tipy ve slovní formě:

1. The Greek lives in house three.
2. There are two houses between the house of the person drinking espresso and the red house.
3. The third house is pink.
4. The person smoking Kools lives directly next to the person drinking coffee.

Houses						
Color:	magenta	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	pink	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	red	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	gray	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	yellow	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Drink:	wine	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	espresso	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	milk	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	coffee	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	tea	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Nationality:	British	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	German	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Swede	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Greek	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Spanish	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Pet:	birds	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	tortoises	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	mice	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	horses	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	butterflies	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Smoker:	Marlboro	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Chersterfields	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Blend	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Pall Mall	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Kools	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Action:		<input type="button" value="Show Mistakes"/> <input type="button" value="Check Result"/> <input type="button" value="Clear All"/> <input type="button" value="Set All"/>				

Obrázek 5: Příklad zadání úlohy na www.mensus.net

5. There is one house between the house with the mice and the gray house on the right.
6. There are two houses between the magenta house and the house the Spanish lives in.
7. The birds live to the left of the person drinking milk.
8. There are two houses between the British and the Chersterfields smoking person.
9. The person smoking Pall Mall lives directly next to the yellow house.
10. There is one house between the German and the Greek on the right.
11. There is one house between the magenta and yellow house on the right.
12. There are two houses between the horses and the person smoking Pall Mall.
13. There is one house between the tortoises and the person drinking tea on the right.
14. The person smoking Blend lives directly next to the magenta house.

Where does everybody live?

Number of houses: (3 – 10)

Attributes:

- Color: Each house has a different color.
- Nationality: The people living in the houses all have a different nationality.
- Drink: Each person has a different favorite drink.
- Pet: Each house is home to a different pet.
- Smoker: Each person prefers a different type of tobacco product.
- Food: Each person prefers a different type of food.
- Trees: A different type of tree grows in front of each house.
- Sport: Each person plays a different sport.
- Flowers: Different flowers grow in front of each house.
- Car: Each person drives a different type of car.

Types of clues:

- The person/animal/plant lives/grows in a given house.
- The person/animal/plant does not live/grow in a given house.
- The person/animal/plant lives in the same house as the other person/animal/plant.
- The person/animal/plant is a direct neighbor of the other person/animal/plant.
- The person/animal/plant is the left or right neighbor of other person/animal/plant.
- There is one house between the person/animal/plant and the other person/animal/plant.
- There is one house between the person/animal/plan and the other person/animal/plant on the left or right.
- There are two houses between the person/animal/plant and the other person/animal/plant.
- There are two houses between the person/animal/plan and the other person/animal/plant on the left or right.
- The person/animal/plant lives left or right from the other person/animal/plant.

K zadání je také vygenerován kód (zde 40FFFE.9E9A962), se kterým je možné později zkontrolovat odpověď jednou vygenerované úlohy.

Na výběr je také velké množství typu vazeb nebo vodítek (clues), ze kterých lze zadání sestavit. Skript interaktivního zadání odpovědi také pomáhá při zpracování a vylučuje chyby.

Jednou z nejznámějších logických hádanek je „ZEBRA PUZZLE“, nebo také se jí někdy říká Einsteinova hádanka, což je logická hádanka, jejímž cílem je na základě 15. tipů určit obyvatele 5 domů. V české verzi například na portálu e-matematika.cz [12]:

Zadání

- Je 5 domů, z nichž každý má jinou barvu.
- V každém domě žije jeden člověk, který pochází z jiného státu.
- Každý člověk pije nápoj, kouří jeden druh cigaret a chová jedno zvíře.
- Žádný z nich nepije stejný nápoj, nekouří stejný druh cigaret a nechová stejné zvíře.

Nápověda

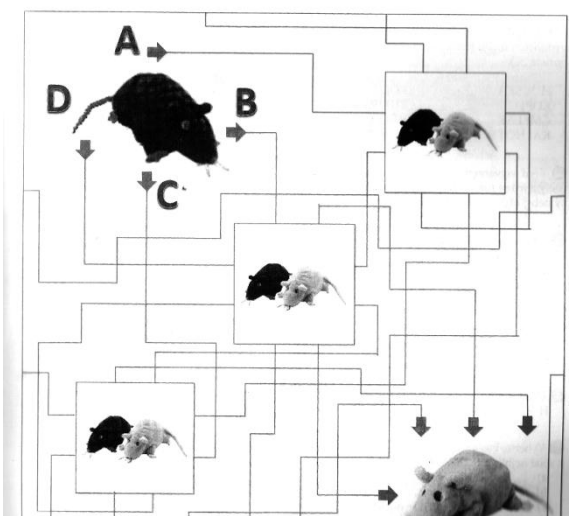
- Brit bydlí v červeném domě.
- Švéd chová psa.
- Dan pije čaj.
- Zelený dům stojí hned nalevo od bílého.
- Majitel zeleného domu pije kávu.
- Ten, kdo kouří PalIMall, chová ptáka.
- Majitel žlutého domu kouří Dunhill.
- Ten, kdo bydlí uprostřed řady domů, pije mléko.
- Nor bydlí v prvním domě.
- Ten, kdo kouří Blend, bydlí vedle toho, kdo chová kočku.
- Ten, kdo chová koně, bydlí vedle toho co kouří Dunhill.
- Ten, kdo kouří BlueMaster, pije pivo.
- Němec kouří Prince.
- Nor bydlí vedle modrého domu.
- Ten, kdo kouří Blend, má souseda, který pije vodu.

Vaším úkolem je přijít na to, kdo chová rybičky.

Tato logická úloha se od předchozích dvou liší v jedné věci. Předchozí úlohy je možné řešit postupně, po jednotlivých krocích, takže vezmete řešitelné pravidlo a aplikujete ho do tabulky. Procházíte postupně všechna pravidla a na konci dostanete vyplněnou tabulku. V tomto případě však dojdete ke kroku 9, a dále již žádné pravidlo nejde aplikovat, jednoduchý algoritmus aplikace se v tomto bodu zastaví. Je tedy nutné zkusit dosadit jednu možnost a zjistit, jestli nebude v rozporu se zbylými pravidly, potom eventuálně zkusit možnost jinou. Tvůrce této logické úlohy obsazoval prvky matice tak, že v matici zůstanou volná místa, nicméně jejich obsazení je vyloučeno kombinatoricky, nikoli pravidly.

2.4 FRAKTÁLOVY JONATHAN

Tato úloha je inspirací pro rekurzivní úlohu SUDRA v praktické části.



Obrázek 6: Fraktálový Jonathan, autor
Vladimír Vančický[14].

„Která z uvedených cest přivede Fraktálového Jonathana k milované Silvě?“
[14] Chodit se smí jenom po cestičkách, ty se mezi sebou nekříží, přes „vzduchoprázdno“ se neskáče. Každá cesta od Jonathana k Silvě musí být jasně a jednoznačně determinována.

Tato úloha je však graficky těžko proveditelná, protože znázornění výsledku by znamenalo iterativní zmenšování obrázku, takže výsledek by byl těžko čitelný. Proto je úloha uzpůsobena nižšímu rozlišení a obrázky se neopakují v sobě ale vedle sebe.

To praktické provedení usnadní.

2.5 JAK SI HRAJÍ LIDÉ

Zatímco předchozí úlohy sloužily k rozvoji matematicko-logické či prostorové inteligence, následující úlohy jsou určeny spíše ke zkoumání chování lidí ve skupině. Tyto úlohy nejsou přímo teorií her, které jsou zpracovatelné matematicky a které jsou schopny najít stabilní řešení – Nashovo equilibrium pro všechny zúčastněné strany.

Jednou z nejobecnějších her z této skupiny je hra „Veřejný statek“ (public goods) [9]. Je zobecněním Vězňova dilematu (prisoner's dilemma). Hráči při ní mohou investovat svoje prostředky do společného projektu, přičemž takto vložené investice jsou zagregovány, poté zhodnoceny, znásobeny a rozděleny rovným podílem mezi investory. Zatím co matematicky mohou investoři v takové hře získat teoreticky zisk rovný násobku zhodnocení investice, prakticky však, je-li proces investování anonymní, díky rovnoměrnému rozdělení výnosů je pro investory výhodné neinvestovat, takže racionálním přístupem není tato hra hratelná a investoři v ní prakticky neinvestují. Možným řešením je deanonymizace investic s možností trestání nebo iracionální motivy jako je vliv autorit či víra.

Tabulka 1: Vývoj investic ve hře public goods

	<p>Každý hráč investuje 1000 jednotek. Součet od všech hráčů je investorem znásoben 2x a poté rozdělen rovnoměrně mezi všechny hráče.</p>
	<p>Pokud však jeden hráč nic neinvestuje, díky agregaci příspěvků a rovnoměrnému rozdělení zisků získá paradoxně nejvíc, v tomto případě 1600 jednotek, což je navíc o 1000 jednotek víc než investující hráči.</p>
	<p>Pokud investuje malý počet hráčů, v tomto případě pouze dva, investující hráči naopak investic ztrácí.</p>

Proto se tyto hry vyskytují v různých provedeních, jako například hra „Diktátor“, „Konec smlouvání“, „Důvěra“, „Veřejný statek“ či „Trestání třetí stranou“, kdy hra není zcela anonymní, hráči mohou altruisticky trestat „černé pasažéry“. [10] Podobných ekonomických her je více druhů, například hry zkoumající jev nazývaný eskalace iracionálních závazků. Ačkoli by se mohlo zdát, že tyto hry vysvětlují jen úzkou část

chování, opak je pravdou. Sociální mozek, který je díky své iracionalitě schopen řešit podobné racionálně neřešitelné úlohy řídí ekonomiku, politiku a sociální vztahy a dá se říci, že je nejsložitější a také nejméně probádanou částí [11].

II. PRAKTICKÁ ČÁST

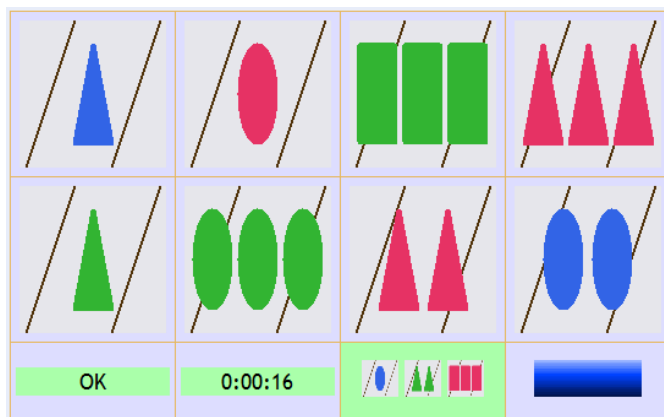
3 VYTVOŘENÉ ÚLOHY

V této části jsou popsány programy, které byly za účelem řešení či generování logických úloh vytvořeny. Bude zde podrobně popsán generátor hry SET, stejně tak řešitel / generátor zebra puzzle, dále ukázány možnosti generátoru úloh na kostce, tvorba rekurzivních úloh a na konci popsán princip offline verze investiční hry „public goods“.

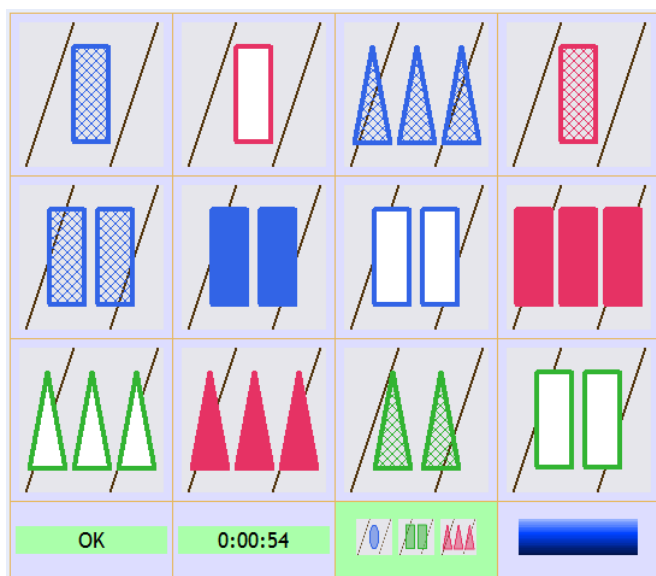
3.1 SET

Tato hra byla původně realizována jako semestrální práce v jazyku PYTHON s využitím wxWidget. Oproti karetní hře bylo nutné ošetřit stavy, kdy na stole mezi 12. kartami není žádný SET. To se při fyzické hře řeší dokládáním 3 karet, dokládání probíhá tak dlouho, dokud SET není nalezen. Na tom, že na stole SET není, se hráči domluví (prakticky to neznamená, že není, ale možná není, nebyl nalezen). V grafickém prostředí by bylo rozšiřování pole karet problematické, navíc by k němu muselo dojít již při vykládání či výměně karet, protože hraje se proti automatu, takže vzájemnou dohodou toto řešit nelze. Tento problém byl vyřešen tak, že jsou vyložené karty prohledány, zda obsahují alespoň jedno řešení, v opačném případě jsou doplňující karty vráceny do balíčku, tento je promíchán a karty opět doloženy. Tím se sice odstranil problém přidávání karet, zůstal ale problém s neurčitým koncem hry. Hra končí, v původní i programové verzi, když není nalezen i po vyložení všech karet žádný set. Jak již bylo uvedeno v teoretické části, může tato situace nastat i při počtu 20. karet na stole, což znamená, že hra končí po 20. kolech. Teoreticky na druhé straně může nastat situace, kdy na stole nezůstane karta žádná, pak by hra končila po 27. kolech. Tento velký rozdíl je ve finální verzi řešen tím způsobem, že počet kol je jednoznačně zadán. Dalším problémem se ukázala složitost hry. Ve fyzické hře proti sobě hrají hráči, přičemž ten nejrychlejší získává body. V programové verzi to znamenalo nastavit rychlost automatu tak, aby simuloval živého protihráče. Toho bylo dosaženo analýzou složitosti zadání, tj. počtem SETů na stole a složitosti jejich nalezení, přičemž SETy obsahující karty se stejnými atributy jsou nalezeny rychleji než SETy s atributy různými. Přesto však z logování vyplynulo, že někteří hráči tempu nestačili vůbec a po několika neúspěšných kolech hru ukončili, zatím co jiní končili opakovaně s plným počtem bodů. Proto ve finální verzi není čas zadán a hráči mají na nalezení libovolný čas, výsledkem je potom čas pro odehrání pevně stanoveného počtu kol. Také byly zrušeny záporné body za chybně označený SET (v karetní hře je za správný SET +3 body, za chybný -1 bod, počítá se tak, že hráči si karty nalezených SETů ponechávají, při chybě

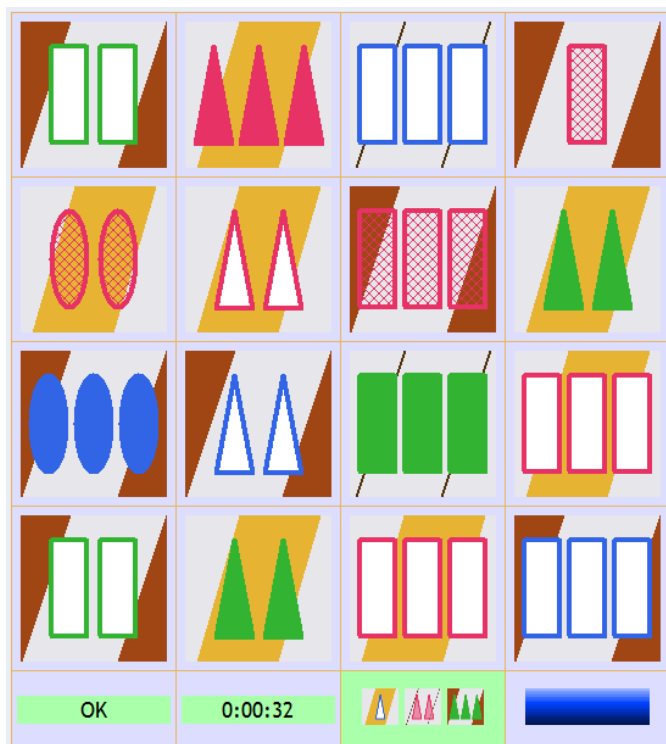
jednu vracejí, karty se sečtou na konci hry) (Obrázek 10). Další změnou bylo rozdělení hry do stupňů složitosti, přičemž složitost je dána počtem atributů karty. V původní verzi jsou atributy 4, pro snadnější verzi je odstraněn atribut výplně, karetní pole je zmenšeno z 4x3 (Obrázek 8) na 4x2 (Obrázek 7), ve složitější verzi byl naopak jeden atribut přidán, je to pozadí karty a hrací pole bylo rozšířeno na 4x4 (Obrázek 9).



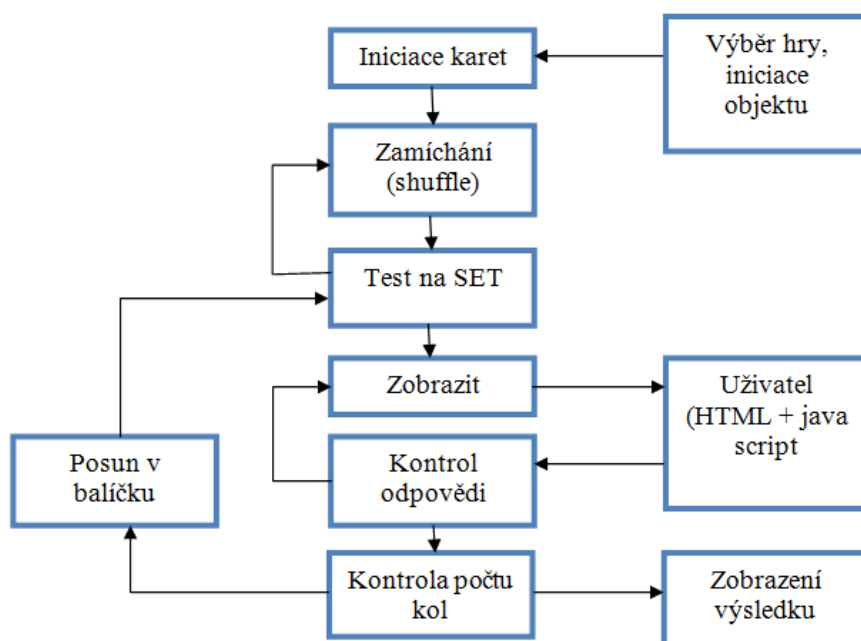
Obrázek 7: Hrací pole 4x2



Obrázek 8: Hrací pole 4x3



Obrázek 9: Hrací pole 4x4



Obrázek 10: Schéma generátoru pro hru SET.

3.2 KOSTKA

Tento program slouží k automatickému vytváření úloh na kostce. Základem programu je generátor náhledu na kostku buď ve formě 3D pohledu nebo na rozvinutý plán kostky. Z těchto pohledů je možné sestavit obrázek celého zadání, ke kterému je pak v HTML vytvořeno zadání a formulář pro odpověď. Může však sloužit i jako generátor zadání pro papírovou podobu, proto jsou generované obrázky čistě černobílé pro zjednodušení tisku či kopírování.

Datovou strukturou kostky je pole s 6. stranami, na kterých se definuje:

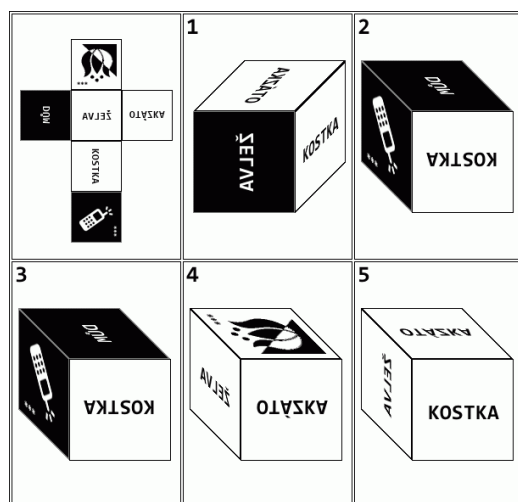
- obsah strany - text / znak / obrázek
- inverzní zobrazení – ano / ne
- rotace – 0 / 90 / 180 / 270°
- zrcadlové zobrazení – ano / ne

Z těchto struktur se pak skládá celý obrázek, který je tvořen 6. pohledy, u každého je uložena struktura kostky a pohled, který se skládá z parametrů:

- pohled – rozvinutý plán / horní pravý / dolní pravý / dolní levý / horní levý
- inverze – ano / ne
- čelní strana – 1-6
- rotace – 0 / 90 / 180 / 270°

Účelem je pak cvičení především prostorové představivosti, například identifikace stran při různých pohledech nebo rotaci. Generátor umožňuje pohled na 6 čelních stran, 4 úhly rotace a 4 pohledy v 3D zobrazení, což dává 96 možností zobrazení pro 3D zobrazení a 24 možností pro pohled na plán. Proto lze provádět změny například:

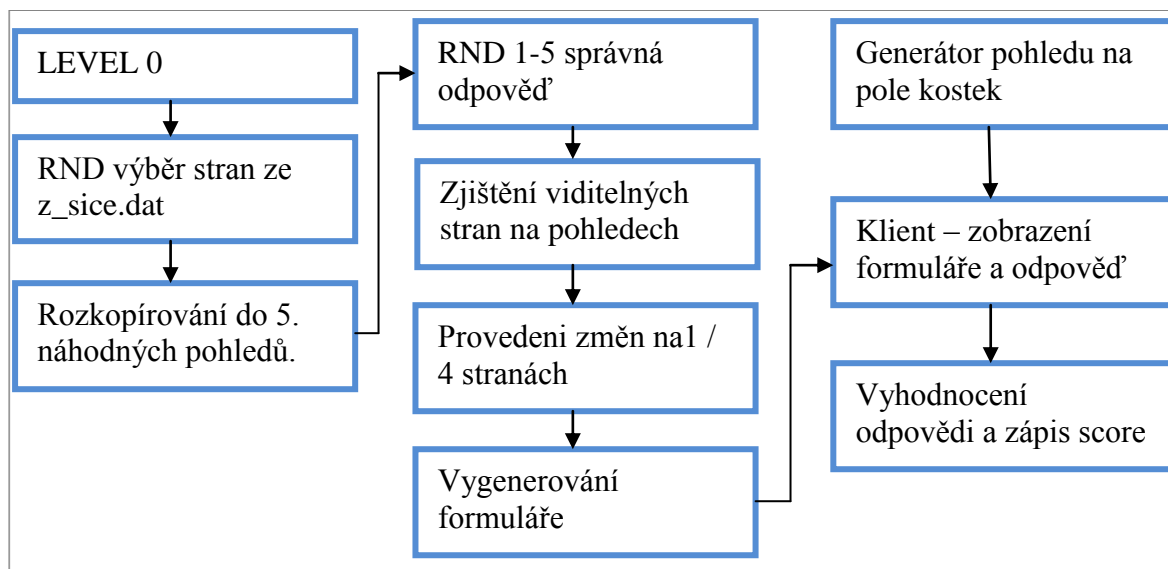
- rotace jedné strany (zobrazovaný znak nesmí být symetrický)
- zrcadlové zobrazení (zobrazovaný znak nesmí být stranově symetrický)



Obrázek 11: Příklad 1 LEVEL 0, chybný je pohled 1, text „ŽELVA“ je invertován

- inverzi
- prohození stran

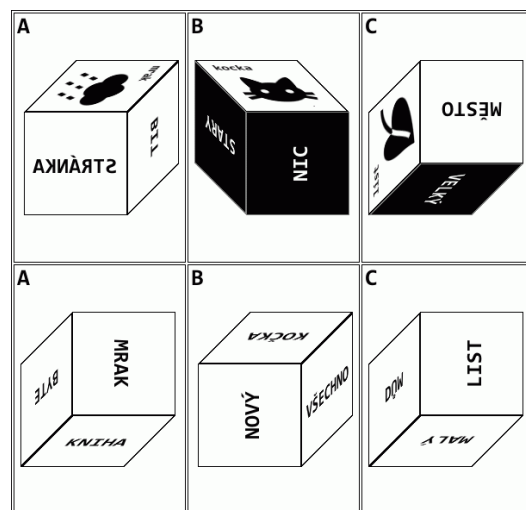
V systému jsou pak aplikovány 2 úrovně složitosti.



Obrázek 12: Schéma generátoru pro kostku LEVEL 0

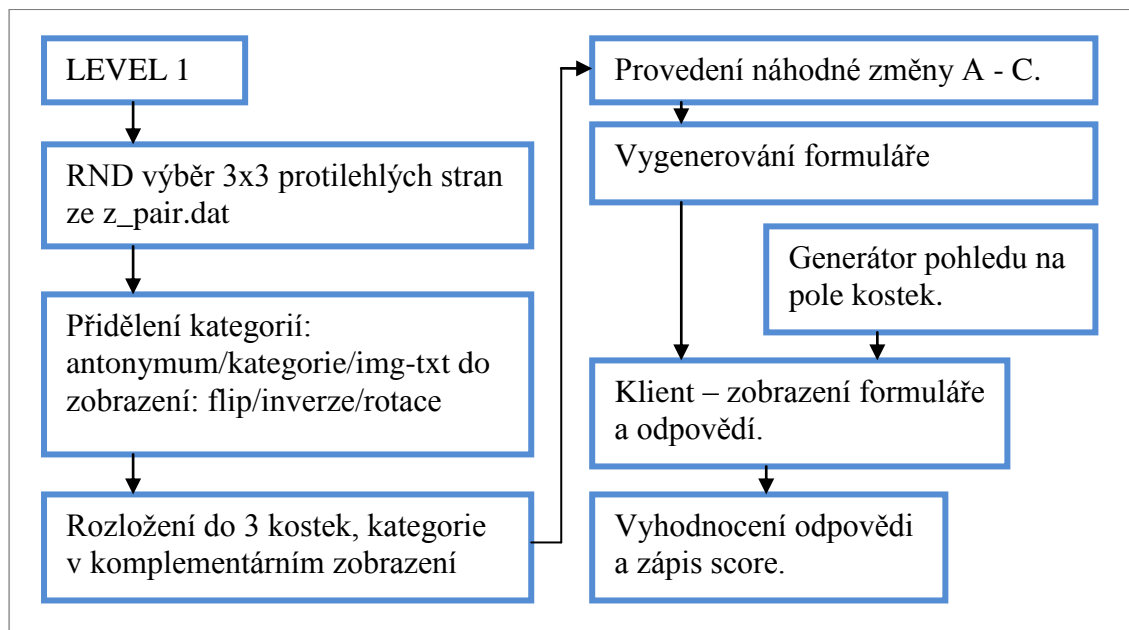
V první (LEVEL 0) je úloha vytvořena tak (Obrázek 11), že se ze seznamu náhodně vybere obsah stran kostky, ta se poté rozkopíruje do všech polí celého zobrazení a vydefinují se jednotlivé pohledy. Ty poté představují různé pohledy na jednu kostku. V těchto pohledech je pak provedena změna na jedné kostce nebo jsou provedeny změny na všech kostkách kromě jedné. Úloha je pak definována jako nalezení pohledu na kostku, který odpovídá / neodpovídá zobrazenému plánu (Obrázek 12).

Nicméně tento generátor byl vytvořen i ke generování složitějších úloh (Obrázek 14), kdy cílem není jenom procvičování prostorové představivosti ale spojení více pohledů na problém. Například různým zobrazením synonym / antonym / kategorií. Takže v druhé úloze (LEVEL 1) jsou vytvořeny 3 kostky (Obrázek 13), chybná je kostka B, strany



Obrázek 13: Příklad 2 LEVEL 1,
chybná je kostka B

"VŠECHNO" a "KOČKA" jsou zaměněny. Strany každé z nich jsou vybrány ze seznamu párů, které budou umístěny protilehle. Protože páry jsou v logickém vztahu (aplikovány páry – kategorie, antonymum, text/obrázek), je možné typ vztahu vyjádřit graficky, tedy rotací, stranovým převrácením (flip) nebo inverzí. Pro vyřešení úlohy je třeba pochopit jak logickou vazbu mezi stranami tak jejich prostorové uspořádání.



Obrázek 14: Schéma generátoru pro kostku LEVEL 1

3.3 ZEBRA PUZZLE

Pro tento typ úloh byl vytvořen program v jazyce PHP, který umožňuje řešení či generování úloh tohoto typu. Obsahuje části pro různé postupy při řešení.

Základní částí úlohy jsou tři matice, které definují stavy a vazby. Další matice a proměnné třídy jsou pomocné, jsou to například názvy polí a atributů, názvy vazeb, matice pro generování vztahů v textové podobě.

První maticí je matice vazeb. V programu jsou implementovány tyto druhy vzájemné vazby:

Tabulka 2: Relace mezi prvky použité v programu "zebra puzzle"

Popis $A=cA[A]$, $B=cB[B]$	Zobrazení vztahu	rel_type	Matice
A má B	$ = $	1	0
A je vlevo od B	$ A B $	2	1
A je vpravo od B	$ B A $	3	-1
A přímo sousedí s B	$?B A ?B $	4	-1,1
A je ob jedno vlevo od B	$ A - B $	5	2
A je ob jedno vpravo od B	$ B - A $	6	-2
A ob jedno sousedí s B	$?B - A - ?B $	7	-2,2
A je ob dva vlevo od B	$ A - - B $	8	3
A je ob dva vpravo od B	$ B - - A $	9	-3
A ob dva sousedí s B	$?B - - A - - ?B $	10	-3,3
A nemá B	$ x $	-1	0
A přímo nesousedí s B vpravo	$ A xB $	-2	1
A přímo nesousedí s B vlevo	$ xB A $	-3	-1
A přímo nesousedí s B	$ xB A xB $	-4	-1,1
A je vlevo od B	$ A ?B ?B... $	51	range(1, (Cols)-1)
A je vpravo od B	$...?B ?B A $	52	range(-(Cols)+1,-1)

Vztahy jsou popsány v matici:

AtributA, PrvekA, AtributB, PrvekB, IndexVazby,(vazba uplatněna, více se neprovádí)

Příklad (var_dump(relations)):

```
array
0 => //vazba 0
  array
    'A' => int 1 //polozka A
    'cA' => int 2 //atribut A
    'B' => int 3 //polozka B
    'cB' => int 1 //atribut B
    'rel_type' => int 51 //51 - A je vlevo od B
```

```
'included' => boolean true //vazba byla uplatnena
```

```
....
```

Samotné pole definují 2 matice, „zebra“ je matice možností, true určuje, že se na daném místě prvek může vyskytovat. False znamená, že přítomnost již byla vyloučena.

Příklad (var_dump(zebra))

```
array
  1 =>
    array
      1 => //prvni atribut
        array
          1 => boolean true //..(prvek může být na tomto místě)
          2 => boolean false
          3 => boolean false //..(prvek nemůže být na tomto místě)
          4 => boolean false
          5 => boolean false
      2 => //druhy atribut
        array
          1 => boolean false
          2 => boolean true
          3 => boolean false
          4 => boolean false
          5 => boolean false
```

```
....
```

Druhou maticí je „solve“ ve které jsou již nalezené pozice prvků. 0 znamená stav neznámý, jiná hodnota znamená číslo prvku na pozici. První matice (číslo domu nebo číslo patra) je již od začátku definovaná, je to ta ke které se vztahují vazby typu „v prvním domě bydlí...“.

Příklad (var_dump(solve)):

```
array
  1 =>
    array
      1 => int 1
      2 => int 2
      3 => int 3
      4 => int 4
      5 => int 5
  2 =>
    array
      1 => int 0 //..(není známo jaký prvek bude na tomto místě)
      2 => int 0
      3 => int 0
      4 => int 5 //..(na tomto místě je prvek 5)
      5 => int 4
```

Tuto matici by bylo možné nahradit dynamickým zjišťováním nalezených prvků (prvek je nalezen na daném místě, pokud je v matici „zebra“ již jen jedna možnost výskytu). Z

důvodů snížení výpočetní náročnosti se však nalezené prvky zapíšou pouze jednou a více se již nevyhledávají.

Úlohy typu zebra puzzle lze řešit několika způsoby. Nejjednodušším je vyzkoušet všechny možnosti, u každé projít všechny pravidla, pokud alespoň jedno z nich nevyhovuje možnost vyloučit. Nevýhodou je velké množství možností, které je třeba prověřit. Počet možností je dán vztahem pro permutaci:

$$P_{(n)} = n!$$

V případě víceatributové úlohy bude navíc nutné otestovat všechny možnosti ve všech řádcích, takže výsledné množství možností bude:

$$P_{(n)} = (n!)^a$$

Jako příklad je zpracována úloha z teoretické části: Dinesmanův bytový problém.

(Dinesman's multiple-dwelling problem [11]). Český překlad:

- Andrea, Broňa, Cecílie, Dana a Eva bydlí v pětipatrovém domě v různých patrech.
- Andrea nebydlí úplně nahoře.
- Broňa nebydlí v přízemí.
- Cecílie nebydlí ani pod střechou ani v přízemí.
- Dana bydlí nad Broňou.
- Eva nebydlí přímo nad ani pod Cecilkou.
- Cecilka nebydlí přímo pod ani nad Broňou.

Vytvoříme novou úlohu o jednom atributu „JMÉNO“ a pěti sloupcích, (atribut „číslování“ je automatický) a zadáme relace:

- REL:0 - JMÉNO Andrea | x | POŘADÍ 5
- REL:1 - JMÉNO Broňa | x | POŘADÍ 1
- REL:2 - JMÉNO Celílie | x | POŘADÍ 1
- REL:3 - JMÉNO Celílie | x | POŘADÍ 5
- REL:4 - JMÉNO Dana | ...?B ?B A | JMÉNO Broňa
- REL:5 - JMÉNO Eva | xB A xB | JMÉNO Celílie
- REL:6 - JMÉNO Celílie | xB A xB | JMÉNO Broňa

Spustíme „PERMUTATOR“, který začne zkoušet kombinace a po nalezení přípustného vypíše:

NALEZENO ŘEŠENÍ

Tabulka 3: Řešení Dinsemanova „bytového“ problému.

PATRO	1	2	3	4	5
JMÉNO	Eva	Broňa	Andrea	Cecílie	Dana

Řešení úlohy uvedené v teoretické části (mensus.net)

HOUSE	1	2	3	4	5
COLOR	red	magenta	pink	yellow	gray
magenta	X	O	X	X	X
pink	X	X	O	X	X
red	O	X	X	X	X
gray	X	X	X	X	O
yellow	X	X	X	O	X
DRINK	coffee	wine	tea	espresso	milk
wine	X	O	X	X	X
espresso	X	X	X	O	X
milk	X	X	X	X	O
coffee	O	X	X	X	X
tea	X	X	O	X	X
NATIONALITY	German	Swede	Greek	British	Spanish
British	X	X	X	O	X
German	O	X	X	X	X
Swede	X	O	X	X	X
Greek	X	X	O	X	X
Spanish	X	X	X	X	O
PET	tortoises	horses	mice	birds	butterflies
birds	X	X	X	O	X
tortoises	O	X	X	X	X
mice	X	X	O	X	X
horses	X	O	X	X	X
butterflies	X	X	X	X	O
SMOKER	Chesterfields	Kools	Blend	Marlboro	Pall Mall
Marlboro	X	X	X	O	X
Chesterfields	O	X	X	X	X
Blend	X	X	O	X	X
Pall Mall	X	X	X	X	O
Kools	X	O	X	X	X

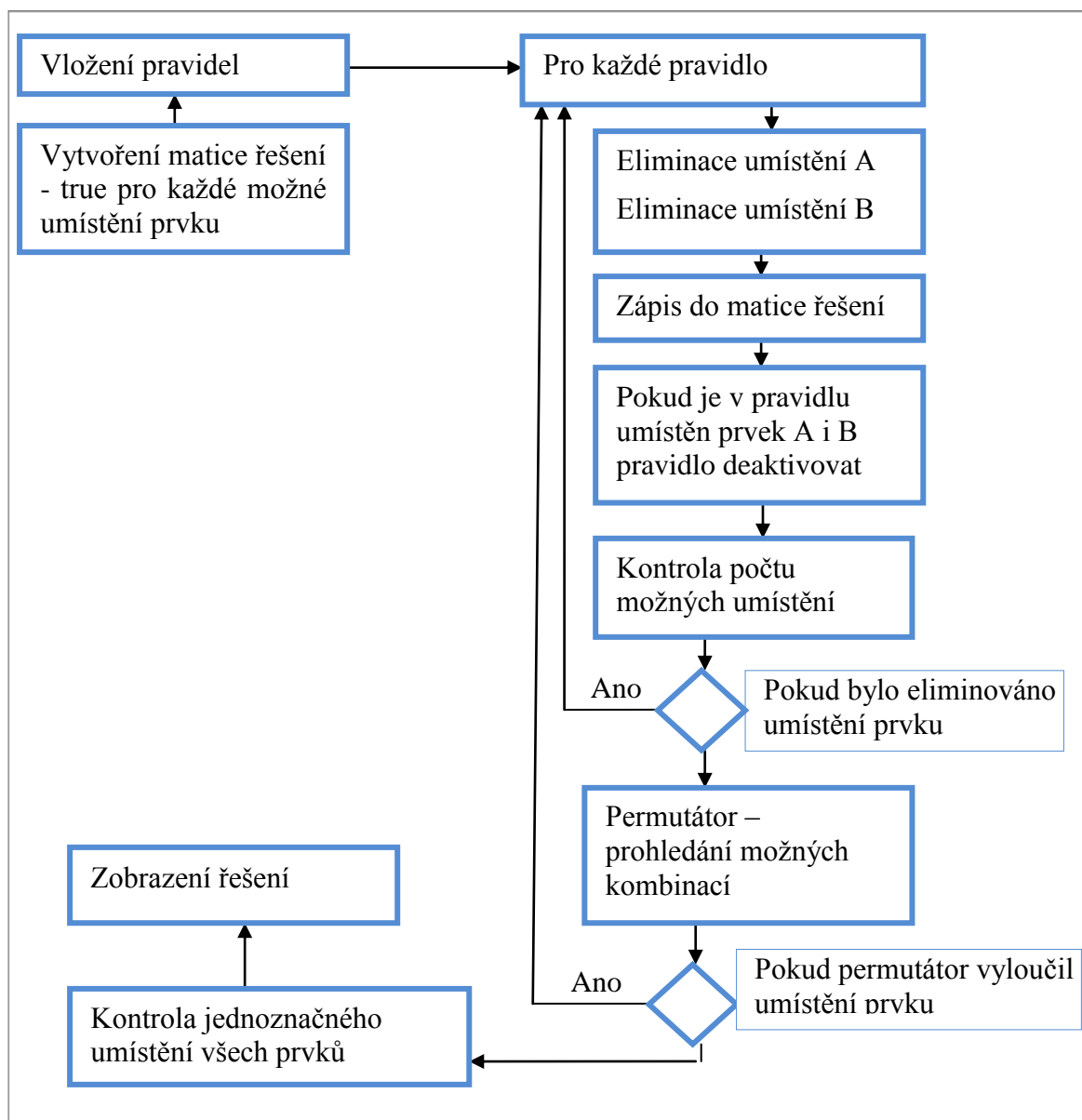
Obrázek 15: Vyřešená úloha z mensus.com

Některé zebra puzzle jsou řešitelné zcela po krocích, eliminováním obsazení jednotlivých polí na základě pravidel (Obrázek 15). Jiná situace je u „Einsteinovy hádanky“, uvedené v teoretické části. Při ní se postupnou eliminací dostaneme až do bodu, kdy žádné další pravidlo nemůžeme uplatnit. V tom případě můžeme použít vylučování možností

prohledáváním existujících variant. Permutátor postupně zkouší u prvků, které dosud nebyly jednoznačně umístěny jednotlivé variace. Umístění prvků, které se neobjevilo v žádné variantě je označeno jako nemožné (false). Poté opět (Obrázek 16) pokračuje vylučování po krocích.

Tímto způsobem je řešeno i „Einsteinovy hádanky“, které proběhne ve 2 krocích (Obrázek 18), (Obrázek 17).

Generování úloh tohoto typu je pak řešeno jako postupné generování jednotlivých vazeb. Ty jsou vybírány tak, že řádků jednotlivých prvků se vybere ten s největším množstvím možných umístění, poté náhodně některý s nižším množstvím možných umístění a u těchto dvou prvků se zjistí úspěšnost eliminací pro jednotlivé typy vazem. Vazby, které způsobí kolizi prvků, jsou vyloučeny a je zařazena vazba s nejvyšším počtem eliminací. Toto probíhá cyklicky, dokud není pro všechny prvky jednoznačně určeno umístění.



Obrázek 16: Schéma řešitele úloh "zebra puzzle"

DUM	1	2	3	4	5
BARVA	zluta	modra	cervena	zelena	bila
cervena	X	X	O	X	X
zelena	X	X	X	O	X
bila	X	X	X	X	O
zluta	O	X	X	X	X
modra	X	O	X	X	X
NARODNOST	Norsko	?	Anglie	?	?
Anglie	X	X	O	X	X
Svedsko	X	X	X	O	O
Dansko	X	O	X	X	O
Norsko	O	X	X	X	X
Nemecko	X	O	X	O	O
NÁPOJ	?	?	mleko	kava	?
čaj	X	O	X	X	O
kava	X	X	X	O	X
mleko	X	X	O	X	X
pivo	X	O	X	X	O
voda	O	O	X	X	O
CIGARETY	dunhil	?	?	?	?
pallmall	X	X	O	O	O
dunhil	O	X	X	X	X
blend	X	O	O	O	X
bluemaster	X	O	X	X	O
prince	X	O	X	O	O
ZVÍŘE	?	kun	?	?	?
pes	X	X	X	O	O
ptaci	X	X	O	O	O
kocka	O	X	O	O	O
kun	X	O	X	X	X
ryba	O	X	O	O	O

Obrázek 18: Řešení "Einsteinovy hádanky" 1

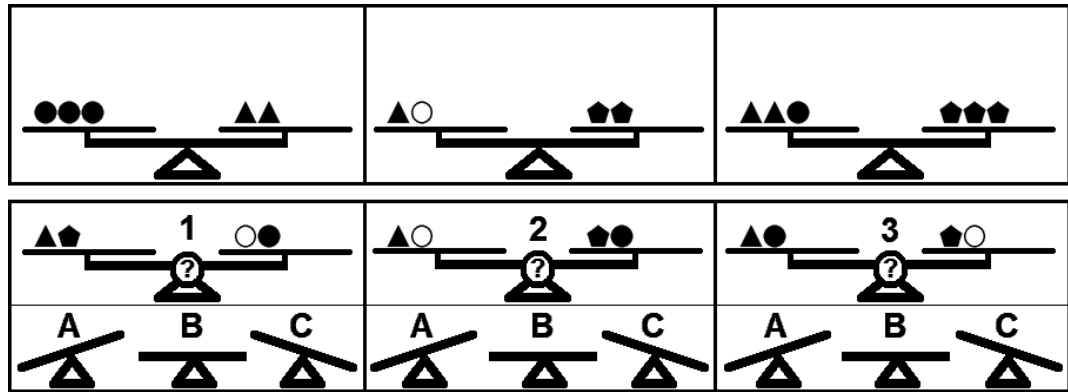
DUM	1	2	3	4	5
BARVA	zluta	modra	cervena	zelena	bila
cervena	X	X	O	X	X
zelena	X	X	X	O	X
bila	X	X	X	X	O
zluta	O	X	X	X	X
modra	X	O	X	X	X
NARODNOST	Norsko	Dansko	Anglie	Nemecko	Svedsko
Anglie	X	X	O	X	X
Svedsko	X	X	X	X	O
Dansko	X	O	X	X	X
Norsko	O	X	X	X	X
Nemecko	X	X	X	O	X
NÁPOJ	voda	čaj	mleko	kava	pivo
čaj	X	O	X	X	X
kava	X	X	X	O	X
mleko	X	X	O	X	X
pivo	X	X	X	X	O
voda	O	X	X	X	X
CIGARETY	dunhil	blend	pallmall	prince	bluemaster
pallmall	X	X	O	X	X
dunhil	O	X	X	X	X
blend	X	O	X	X	X
bluemaster	X	X	X	X	O
prince	X	X	X	O	X
ZVÍŘE	kocka	kun	ptaci	ryba	pes
pes	X	X	X	X	O
ptaci	X	X	O	X	X
kocka	O	X	X	X	X
kun	X	O	X	X	X
ryba	X	X	X	O	X

Obrázek 17: Řešení "Einsteinovy hádanky" 2

- REL:0 - ~~BARVA BRIT | = | STÁT ČERVENÁ~~
- REL:1 - BARVA ŠVÉD | = | ZVÍŘE PES
- REL:2 - BARVA DÁN | = | NÁPOJ ČAJ
- REL:3 - ~~STÁT ZELENÁ | A B | STÁT BÍLÁ~~
- REL:4 - ~~STÁT ZELENÁ | = | NÁPOJ KÁVA~~
- REL:5 - CIGARETY PALLMALL | = | ZVÍŘE PTÁCI
- REL:6 - ~~STÁT ŽLUTÁ | = | CIGARETY DUNHIL~~
- REL:7 - ~~POŘADÍ 3 | = | NÁPOJ MLÉKO~~
- REL:8 - ~~BARVA NOR | = | POŘADÍ 1~~
- REL:9 - CIGARETY BLEND | ?B A ?B | ZVÍŘE KOČKA
- REL:10 - ZVÍŘE KONĚ | ?B A ?B | CIGARETY DUNHIL
- REL:11 - CIGARETY BLUEMASTER | = | NÁPOJ PIVO
- REL:12 - BARVA NĚMEC | = | CIGARETY PRINCE
- REL:13 - ~~BARVA NOR | ?B A ?B | STÁT MODRÁ~~
- REL:14 - CIGARETY BLEND | ?B A ?B | NÁPOJ VODA

3.4 Váhy

Tato úloha je ryze matematická. Jedná se o úlohu, kde jsou čísla a proměnné nahrazeny předměty a fyzikálními zákony. Cílem je pochopit a naučit se vyřešit problém řešením soustavy rovnic o více neznámých.



Obrázek 19: Matematický úkol: určete, jak se vychýlí spodní váhy 1-3.

Jsou zobrazeny 2x3 váhy s různými předměty. 3 váhy v horní řadě představují soustavu tří rovnic o čtyřech neznámých, kdy obě strany rovnice se rovnají. Úkolem je vyřešit tyto rovnice a určit, jak se budou chovat tři spodní váhy s otazníkem, tedy zda-li budou strany v rovnováze nebo nerovnováze.

$$3A = 2B; B + C = 2D; 2B + A = 3D$$

Můžeme také využít matlab/octave a použít výpočet koeficientů rovnic o více neznámých pomocí matic:

$$X = M^{-1} * V$$

```
octave:1> m=[3 -2 0;0 1 1;1 2 0] //matice m
m =
    3   -2    0
    0    1    1
    1    2    0

octave:2> v=[0;2;3] //matice vysledku
v =
    0
    2
    3

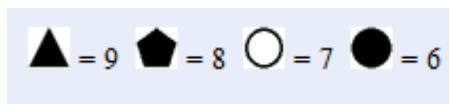
octave:3> x=inv(m)*v //vypocet koeficientu
x =
    0.75000
    1.12500
    0.87500

octave:4> x*det(m) // vynasobeni determinantem
```

```
ans =
-6 //A
-9 //B
-7 //C
```

```
//a D = 8
```

Takže A=6; B=9, C=7 a D=8.



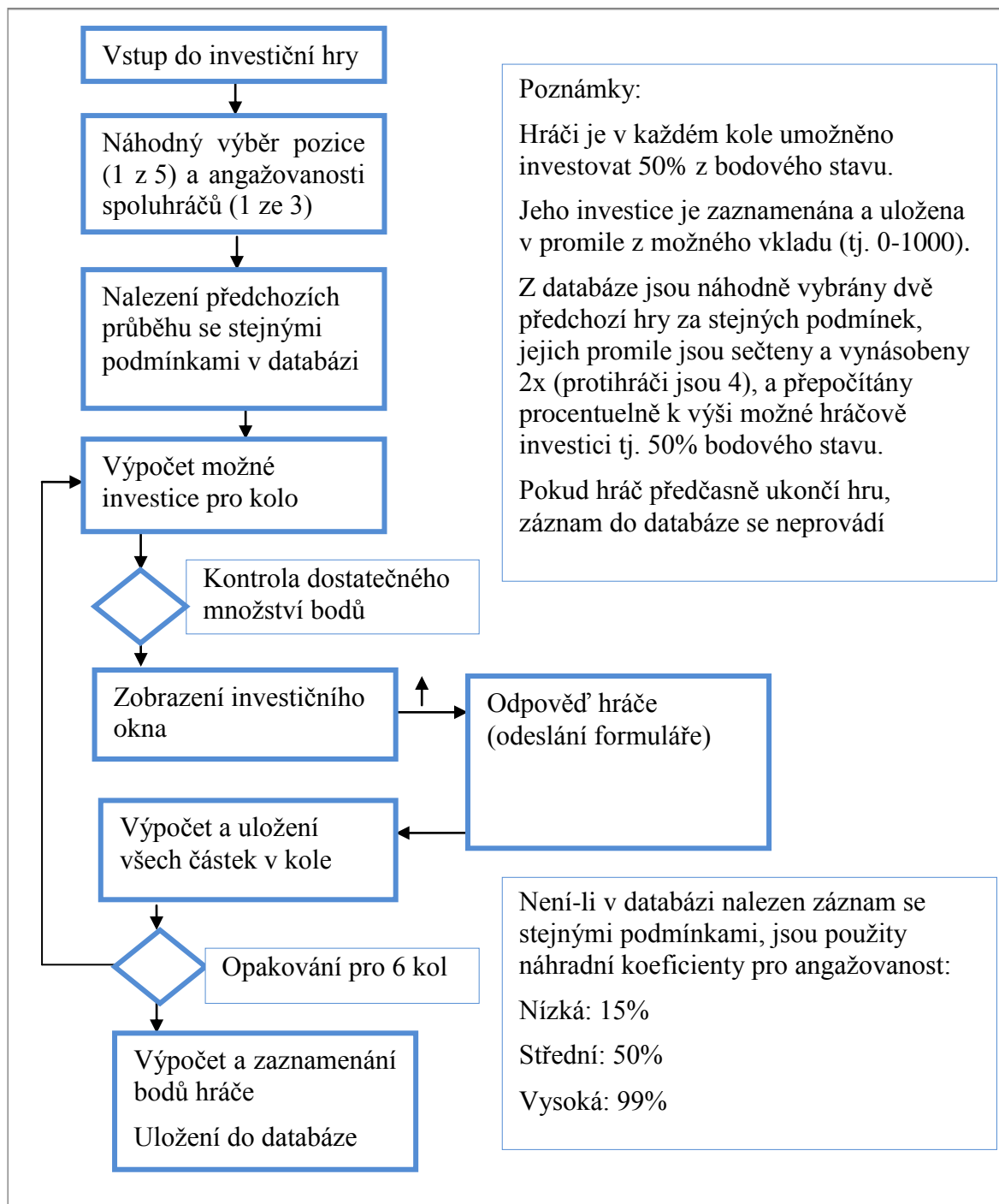
Obrázek 20: Zobrazené řešení
z předchozího obrázku.

Generování úlohy probíhá tak, že náhodně vyberou 4 koeficienty (z hodnot 2-9), k nim se přiřadí náhodně 4 různé geometrické tvary a na horních vahách se pomocí funkcí gcf a lcm (skripty ze z_math.php) a vyhledávání kombinací spočítají počty prvků na ramenech vah o „rozumném“ počtu, tj. aby počet jednoho prvku nepřekročil cca. 4, z důvodu snadného výpočtu. Samozřejmě by bylo možné naplnit trojúhelníkovou matici (která je v podstatě jádrem této úlohy) náhodnými čísly (počty prvků), nicméně takováto matice by sice měla vždy řešení matematické, ovšem fyzikálně by záporné koeficienty (záporná hmotnost prvků) byla chybou.

3.5 Investor

Hlavní myšlenkou této aplikace bylo umožnit získání údajů o chování lidí. Tyto informace jsou běžně získávány při online studiích, kdy skupiny respondentů, obvykle studentů, hrají investiční hry o skutečné či fiktivní peníze proti sobě či výzkumníkovi. Důležité u těchto her je to, že je hrají živí lidé proti jiným živým lidem. Jelikož jejich chování nejsme schopni simulovat (to je cílem tohoto výzkumu), nemůžeme proti nim postavit robota, který by online hru hrál. Možností je tedy ukládat data z jejich chování a z tohoto balíku dat jim překládat online dříve získané odpovědi spoluhráčů. Předpokladem tedy je, že pokud chování lidí vykazuje určité znaky, bude průběh her časem konvergovat do stabilního stavu bez ohledu na počáteční stav, který jsme schopni na začátku pouze kvalifikovaně odhadovat, tedy nastavit iniciační hodnoty.

Prakticky se jedná o matematickou simulaci 6 investičních kol 5 investorů, přičemž zhodnotitel vklady zdvojnásobuje. Jako iniciační hodnoty je náhodně vybírána jedna ze třech variant (vysoká / střední / nízká) angažovanosti spoluhráčů, které jsou postupně nahrazovány údaji od živých hráčů. Výsledky jsou zaznamenávány do databáze. Z ní jsou poté v dalších hrách náhodně vybírány předchozí průběhy se stejnými vstupními podmínkami a použity jako odpovědi protistrany.



Pro lepší pochopení je u investičního formuláře zobrazena animace, která zobrazuje průběh toku investic, jejich zhodnocení a rozdělení zpět hráčům. Také jsou zobrazeny podrobné propočty o investicích, ziscích a celkových ziscích a nápověda na rozcestníku. Bez ohledu na to si je autor vědom, že řada hráčů nepochopí hloubku tohoto sociálně-ekonomického problému a bude investovat spíše náhodně, což celkovou validitu experimentu může narušit.

3.6 SUDRA

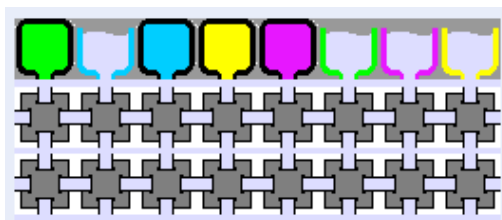
Tato úloha by měla procvičit chápání rekurze.

Jedná se o grafickou úlohu, ve které jsou v horní řadě nádrže s barvou a nádoby, do kterých by

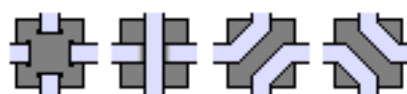
měla být barva napuštěna (Obrázek 23). Pod nimi je pole potrubí. V tomto poli je možné v každém čtverci změnit průběh potrubí ve 4 možnostech (Obrázek 22). Změna průběhu se provádí

klepnutím na prvek, typ se cyklicky mění. Protože průběhů ke zdárnému propojení nádrží s nádobami je málo, musí se pro vyřešení úlohy použít rekurze potrubního pole. Celé pole, tak jak jej uživatel nastaví (Obrázek 21), je rekurzivně opakováno pod sebou, to jest, že spodní výstupy se stanou vstupy v nové iteraci atd. Nastavení pole je řešeno pomocí javascriptu na klientském PC. Po nastavení pole je provedena kontrola správnosti. Ta se provádí na straně serveru, provádí se kontrola průběhu formou simulace průchodu, jsou zaznamenány průběhy a chybné stavy. Z těchto průběhů je vygenerován obrázek. Pokud bylo nastavení správné, budou nádoby zaplněny správnou barvou.

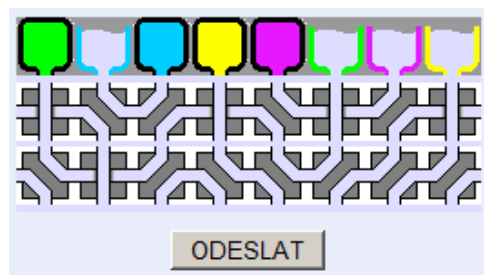
Praktické řešení generování obrázku a kontroly funkce je provedeno simulací průběhu „toku“ po jednotlivých potrubních uzlech směrem z nádrží. Je kontrolováno opuštění pole na bocích nebo po definované rekurzi spodního pole (dole), dále je jako chyba označeno propojení dvou nádrží nebo napojení nádrže na špatnou nádobu. Z této simulace jsou prováděny záznamy o „průchodu“ či ukončení. Z tohoto záznamu je pak ve



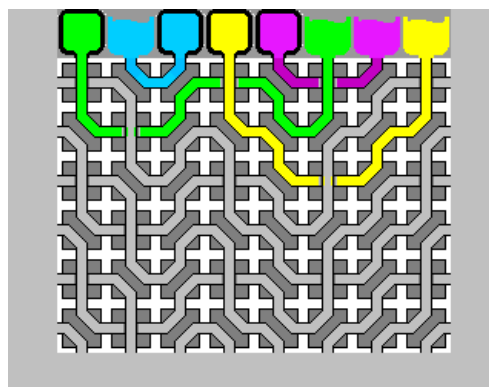
Obrázek 23: Vstupní okno úlohy
SUDRA



Obrázek 22: Typy průběhu
potrubí

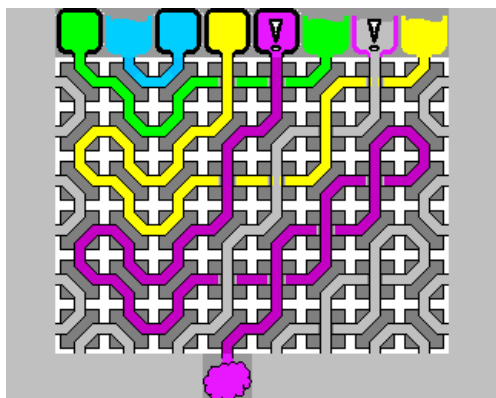


Obrázek 21: Nastavení pole před
odesláním

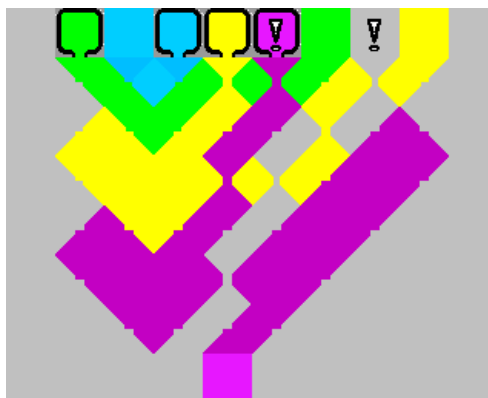


Obrázek 24: Příklad správného
zapojení

skriptu pro generování obrázku vyplňováno pole barevnými prvky s průhledností (Obrázek 25), které jsou skládány přes sebe a vytváří tak podklad průběhu cest. Tento hrubý obrázek je poté překryt polem potrubních uzlů a nádobami (Obrázek 26), které mají v cestách nastaveny průhlednost. Tímto řešením byly nahrazeny pokročilejší grafické funkce, které by umožňovali vybarvování nebo jemnou manipulaci s obrázkem, a které by ve skriptovacím jazyce PHP probíhali neúměrně dlouho.



Obrázek 26: Příklad chybného propojení



Obrázek 25: Příklad chybného propojení bez krycí vrstvy

4 PRAKTICKÉ PROVEDENÍ

V této části je popsáno praktické provedení, detaily řešení pro konkrétní nasazení a zkušenosti s provozem. Jsou také popsány náměty pro další zpracování.

4.1 Platforma

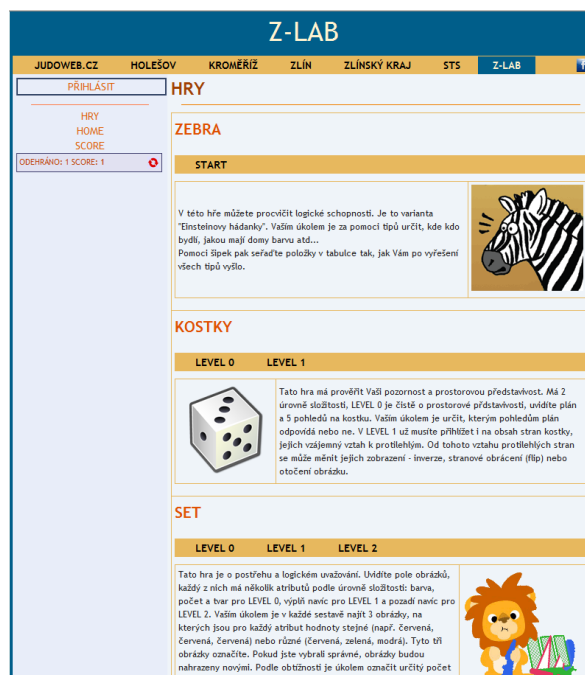
Všechny aplikace jsou napsány v jazyce PHP, pouze generátor obrázků je napsán v jazyce PYTHON s použitím wxWidget. Aplikace jsou spuštěny na serveru jako součást hostingu spolu s několika dalšími portály (Obrázek 27). Aplikace jsou vytvořeny tak, aby nepotřebovaly ke své činnosti žádnou speciální podporu či knihovny, proto je možné jednoduše přenést na jiný hosting. Pouze logování je bez funkční databáze komplikované a proto využívá stávajícího připojení k databázi

MySQL. V systému nejsou použity rozšířené knihovny GD2 či balíček pro práci s obrazovými daty Imagic, což vývoj sice zkomplikovalo, nicméně tyto knihovny či balíčky nejsou standardní součástí XAMP, proto nejsou vyžadovány. Pro účely ladění jsou skripty spouštěny na EasyPHP-5.3.5.0 na stroji s Windows 7.

K uchování hodnot proměnných využívají jednotlivé aplikace globálních proměnných `$_SESSION[„název aplikace“]` do kterých jsou pomocí PHP funkce „serialize/unserialize“ ukládány objekty s hodnotami.

4.2 Možnosti zpracování

Kromě dvou interaktivních úloh – SET a Investor – je možné všechny skripty využít pouze ke generování obrázků s úlohami, které pak mohou být použity v papírové podobě nebo jako součást jiných elektronických médií. Obrázky jsou proto černobílé, aby byl jednodušší jejich eventuelní tisk či kopírování. V případě problému s provozem XAMP serveru je



Obrázek 27: Provozované aplikace na webovém portálu

možné úlohy využít pouze k vygenerování většího množství zadání a ty pak použít v jiné existující „testové“ aplikaci jako vícenásobnou možnost pro zadání otázky.

4.3 Zabezpečení

Generování úloh je výhodné v tom, že nemůže dojít zneužití výsledků zjištěných někým jiným či pod jinou identitou. To bývá například problém u vícekolových soutěží, kdy si účastník zjistí výsledky od jiného účastníka. Do vyšších kol se potom dostávají nekvalitní uchazeči, kteří poté ve vyšších kolech neuspějí. Tomu je možné předejít právě tím, že otázky jsou generovány a jsou s vysokou pravděpodobností vždy odlišné. Alternativou je vygenerování většího množství zadání, které je pak náhodně vybíráno. Toto lze použít, pokud generování úlohy je složité na výkon serveru a při vícenásobném souběžném požadavku by mohlo dojít k přetížení serveru nebo dlouhé časové odezvě.

4.4 Bodování

Každá z aplikovaných úloh funguje samostatně, nicméně započítávání výsledků mají společné. To je zabezpečeno přidáním modulu „score“ do stávajícího webového portálu. Všechny úlohy odesílají výsledky do této aplikace přístupné přes funkci „add_score“, samotný modul poté slouží k zobrazení výsledků jak v banneru tak na samotné stránce. Jednotlivé úlohy posílají do aplikace „score“ jednak počet bodů (kladný či záporný), ale také bližší údaje o typu úlohy a průběhu hry. Tyto údaje jsou poté také ukládány do databáze pro pozdější statistické využití. Přístup do databáze nemají tedy samostatně aplikace úloh ale jen modul „score“. Tyto údaje jsou pak uchovány v globální proměnné „\$_SESSION“. Platnost této proměnné je dána životností session na serveru, což je dáno nastavením XAMP na hostingu, v tomto případě 12 minut. Protože však použitý webový portál používá prvek pro refresh, nedojde ani v případě, že doba odpovědi přesáhne 12 minut k vymazání proměnných session a tím i vynulování score a ztrátě uložených objektů. K tomu však dojde v případě, že klientský počítač bude na dobu delší než 12 minut odpojen od Internetu, vypnut nebo zavřeno okno webového portálu.

Ačkoli započítávání výsledků proběhne okamžitě po jejich výpočtu metodou add_score do modulu „score“, aby se tato změna také ihned zobrazila v okně hostujícího webového portálu, je nutné zabezpečit automatický „reload“ po změně score. V opačném případě by bylo nutné výsledky aktualizovat ručně pomocí refresh okna. Automatické načtení je zajištěno pomocí javascript v otevíraném pop-up okně aplikace úlohy:

v HTML kódu stránky

```
<body onbeforeunload='refreshAndClose();'>
```

javascript v aplikaci úlohy „uloha“.js

```
function refreshAndClose() {  
    if(opener_refresh){  
        window.opener.location.reload(true);  
    }  
}
```

a aktivování funkce „refreshAndClose()“ v posledním okně úlohy. (některé úlohy probíhají interaktivně, výsledek je však započítán až při posledním zobrazení. „Reload“ otevírajícího okna „opener“ při každé interakci by byl zbytečný). Aktivace „reload“ v posledním okně přidaným javascript kódem

```
<SCRIPT LANGUAGE='JavaScript'>opener_refresh = true;</script>
```

Tímto způsobem je řešeno bodování ve všech úlohách.

4.5 Další náměty

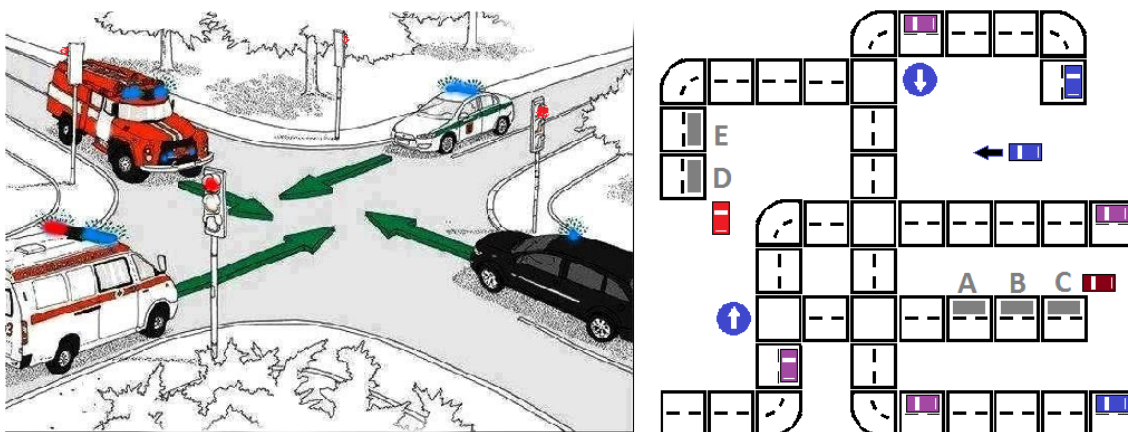
Pro další realizaci zůstali například tyto dva zatím nealgoritmizované dva nápady:

4.5.1 Triplikátor

Je obdobou hry INVESTOR avšak s jiným průběhem. Hráči jsou dva, jeden skutečný, druhý opět simulovaný počítačem. Hráči si navzájem posílají nějakou částku (může být stanovená pevně např. na $\frac{1}{4}$ bodového stavu), cestou k protihráči se tato částka ztrojnásobí. Protihráč může libovolnou částku poslat zpět, zbytek si ponechá. Hra může, ale nemusí mít pevně stanovený počet kol a končit náhodně. Dle toho také hráči pravděpodobně budou volit strategii. Tato hra nabádá ke strategii „oplátka za oplátku“.

4.5.2 Deadlock

Cílem této hry je pochopit „deadlock“, tedy stav, kdy aktéři nemůžou pokračovat v činnosti, jelikož si navzájem blokují prostředky (v tomto případě průjezd křižovatkou). Řešením může být přednost vozidel, v tomto případě je třeba odstranit hrozící stav tak, že se vhodným rozmístěním aut (červeného a hnědého) zpozdí ostatní tak, že se na křižovatce nesejdou najednou.



Obrázek 28: Deadlock.

Úkol: umístěte hnědé a červené auto na pozice A,B,C a D,E tak, aby nedošlo k uváznutí na křižovatkách. (Poděkování neznámému autorovi za pěkný a inspirativní obrázek vlevo).

ZÁVĚR

Hlavní cílem této práce je vytvořit soubor úloh a testů, které mohou sloužit v on-line verzi pro procvičování matematické, logické a prostorové představivosti. Mohou také sloužit i jako součást on-line her, případně generovat zadání do tištěné podoby pro zařazení do matematicko-logických soutěží. Programy jsou vytvořeny v jazyku PHP, který je široce podporovaný a servery LAMP s podporou PHP jsou nejrozšířenějšími systémy nabízenými hostingovými firmami. Pro lokální práci lze využít velkého množství „XAMP serverů“ umožňující práci na vlastním PC jak pro vývoj, tak pro generování úloh. Skripty jsou psány otevřeně a nezávisle na knihovnách či frameworkcích, takže umožňují úpravy a doplňování stávajících generátorů o nové funkce či přizpůsobit novému nasazení bez studia použitých funkcí a služeb. Naopak tato nezávislost na frameworkcích a knihovnách má za následek někdy složitější kód či delší běh skriptu. Kromě samotných komentovaných skriptů jsou v této práci i diagramy, tabulky a slovní popis generátorů, které mohou sloužit k rychlejšímu pochopení jejich principu při případných úpravách či rozšiřování. Důležitou součástí je i logování přístupů a řešení úloh, které mohou sloužit k vyhodnocování jejich složitosti.

Sekundárním cílem je umístění hry INVEST do prostředí, které by umožňovalo smysluplný provoz tohoto experimentu. Smyslem této ekonomické hry není vlastní získávání bodů, ale sledování chování lidí v této investiční hře. Tato hra není samostatně hratelná, proto byla „přilepena“ k ostatním na rozcestník. Jedná se v podstatě o využití šance pro její umístění. Tato aplikace je experimentem bez autorových předchozích zkušeností s podobným sběrem dat, a proto se autor nepokouší odhadovat výsledek či jeho validnost a zpracovatelnost do použitelného závěru. Nicméně i proto je chování hráčů podrobně logováno.

Třetím cílem je nahrazení současné „zábavné“ sekce na provozovaných komunitních stránkách, kterou někteří označovali za příliš složitou a jiní za nudnou.

Závěrem autor apeluje na zájemce o využití přiložených skriptů, aby je nezneužili k útokům na tyto či zahraniční servery, ačkoli by některé funkce v nich obsažené mohly být k takovému účelu použity.

CONCLUSION

The main aim of this work is to create a set of tasks and tests, that can be used in the on-line version for practicing maths, logics and spatial imagination. They can also serve as part of the on-line games, or generate input into printed form for inclusion in mathematical logic competitions. Programs are created in PHP language, which is widely supported. LAMP servers with PHP support are the most widely offered by hosting companies. For local work can take advantage of the large number of "XAMP servers" enabling work on your own PC for both development and generate jobs. Scripts are written openly and independently of libraries and frameworks and they allow modification and completion of existing generators of new functions or adapt them to the new deployment without learning used functions and services. Conversely the independence of frameworks and libraries has resulted in ever more complex code or a longer time for running script. Beside the scripts are annotated in this work as well as charts, tables, and verbal descriptions of generators. That can be used to accelerate the understanding of the principle to any modifications or enhancements. Another important part is the logging of access and problem solving, which can be used for evaluating of their complexity.

The secondary aims is the location of the INVEST game in an environment that would allow meaningful operation of this experiment. This game has not been created for scoring points, but for monitoring of behavior of people who are playing economic games. This game is not playable separately, so it was "glued" to the others on the main page. It is basically the using of a chance for its location. This application is an experiment without the author's prior experience with similar data collection. Therefore the author does not attempt to predict the outcome or its correctness and workability into a usable conclusion. Therefore, the behavior of players is logged in detail.

The third aims is to replace the current "fun" section on the community site operated by some calling it too complicated and boring for others.

Finally, the author appeals to those who are interested in using the attached scripts not to abuse them for attacking this or another server, even though some of the functions contained here might be used for such purpose.

LITERATURA A INFORMAČNÍ ZDROJE

- [1] **Gardner, Howard.** *Intelligence Reframed: Multiple Intelligences for the 21st Century*, Basic Books. 2000. 978-0-465-02611-1.
- [2] **Gardner, Howard.** *Multiple Intelligences: The Theory in Practice*, Basic Books,. 1993. 046501822X.
- [3] *Howard Gardner's Theory of Multiple Intelligences.* **Helding, L.** 2009, Sv. Journal of Singing 66 (2): 193–199.
- [4] **Cihelníková, Michaela, a další.** O teorii inteligencí Howarda Gardnera (prezentace pro psychologii osobnosti UJEP). *Rozmanité Inteligence*. [Online] UJEP, 25. 4 2011. [Citace: 12. 4 2013.] <http://rozmaniteinteligence.blogspot.cz/>.
- [5] **Fisher, Robert.** *Učíme děti myslet a učit se*. místo neznámé : PORTÁL, 2011. ISBN 978-80-262-0043-7.
- [6] **Falco, Marsha Jean.** The SET® Game Company. *SET*. [Online] Set Enterprises, 2011. [Citace: 2. 4 2013.] http://www.setgame.com/set/main_page.htm.
- [7] **Davis, Benjamin Lent; MacLagan, Diane.** THE CARD GAME SET. *Department of Mathematics, The State University of New Jersey*. [Online] [Citace: 12. 3 2013.] <http://www.math.rutgers.edu/~macLagan/papers/set.pdf>.
- [8] **Koltán, I., Koltán, P. a Vittová, K.** *Testy studijních předpokladů a základy logiky - 1. díl*. Brno : Institut vzdělávání SOKRATES s.r.o., 2007. 978-80-86572-38-3.
- [9] **M., Zapletal.** *Kniha hlavolamů*.
- [10] **Mol, Mike.** Dinesman's multiple-dwelling problem. *Rosetta Code*. [Online] 8. 8 2010. [Citace: 20. 3 2013.] http://rosettacode.org/wiki/Dinesman%27s_multiple-dwelling_problem.
- [11] **Křivka, Zbyněk.** Logické úlohy - vyzkoušej svůj intelekt. *Zbyněk Křivka*. [Online] 2005. [Citace: 20. 4 2013.] http://www.fit.vutbr.cz/~krivka/oldpersonal/logicke_ulohy.htm.
- [12] mensus.net - Logic puzzles. *Logic puzzles*. [Online] mensus.net, 2013. [Citace: 10. 1 2013.] <http://www.mensus.net/brain/logic.shtml>.
- [13] **Husar, Petr.** Einsteinova hádanka. *e-matematika.cz*. [Online] e-matematika.cz. [Citace: 10. 1 2013.] <http://www.e-matematika.cz/hadanky/03-einsteinova-hadanka.php>.

- [14] **Vančický, Vladimír.** Fraktálový Jonathan. *Mensa*. XX, 2013, Sv. 3.
- [15] **Koukolík, František.** *ZVÍŘE POLITICKÉ / Eseje o lidské nátuře*. Praha : Galen, 2012. 978-80-762-890-2.
- [16] **Koukolík, František.** *Jak si lidé hrají?* Praha: Radioservis, 2009. 978-80-86212-56-2.
- [17] **Koukolík, František.** *Sociální mozek*. Praha : Karolinium, 2006. 80-246-1242-9.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

- PHP** Hyper text preprocesor – skriptovací programovací jazyk spuštěný na serverové straně.
- LAMP** Spojení Linux (operační systém), Apache (webový server), MySQL (databáze) a PHP (serverový skriptovací jazyk, případně Perl nebo Python).
- XAMP** Spojení Apache/MySQL/PHP může být provozováno i na nelineuxovém stroji. Proto pokud není specifikováno, lze označit jako XAMP. Problémem můžou být rozdílné služby poskytované operačním systémem.
- GD** Graphics Draw, knihovna pro PHP poskytující práci s obrázky.

SEZNAM OBRÁZKŮ

Obrázek 1: SET - příklady správné / nesprávné sestavy.	14
Obrázek 4: Příklad: Na kterých číslech jednotlivé kostky stojí?[8]	15
Obrázek 2: Kostky - příklad.....	15
Obrázek 3: Kostky - příklad.....	15
Obrázek 5: Příklad zadání úlohy na www.mensus.net	17
Obrázek 6: Fraktálový Jonathan, autor Vladimír Vančický[14].....	20
Obrázek 7: Hrací pole 4x2	25
Obrázek 8: Hrací pole 4x3	25
Obrázek 9: Hrací pole 4x4	26
Obrázek 10: Schéma generátoru pro hru SET.	26
Obrázek 11: Příklad 1 LEVEL 0, chybný je pohled 1, text „ŽELVA“ je invertován	27
Obrázek 12: Schéma generátoru pro kostku LEVEL 0	28
Obrázek 13: Příklad 2 LEVEL 1,.....	28
Obrázek 14: Schéma generátoru pro kostku LEVEL 1	29
Obrázek 15: Vyřešená úloha z mensus.com	33
Obrázek 16: Schéma řešitele úloh "zebra puzzle"	35
Obrázek 18: Řešení "Einsteinovy hádanky“ 1	36
Obrázek 17: Řešení "Einsteinovy hádanky“ 2	36
Obrázek 19: Matematický úkol: určete, jak se vychýlí spodní váhy 1-3.....	37
Obrázek 20: Zobrazené řešení z předchozího obrázku.....	38
Obrázek 21: Nastavení pole před odesláním	40
Obrázek 22: Typy průběhu potrubí.....	40
Obrázek 23: Vstupní okno úlohy SUDRA	40
Obrázek 24: Příklad správného zapojení	40
Obrázek 25: Příklad chybného propojení bez krycí vrstvy.....	41
Obrázek 26: Příklad chybného propojení	41
Obrázek 27: Provozované aplikace na webovém protálu	42
Obrázek 28: Deadlock.	45

SEZNAM TABULEK

Tabulka 1: Vývoj investic ve hře public goods	21
Tabulka 2: Relace mezi prvky použité v programu "zebra puzzle"	30
Tabulka 3: Řešení Dinsemanova „bytového“ problému.....	33

SEZNAM PŘÍLOH

P I – VÝPISY ŘEŠENÍ ÚLOH

P II - UKÁZKY ZDROJOVÝCH KÓDŮ KLÍČOVÝCH ČÁSTÍ PROGRAMŮ

P III - UKÁZKY GENEROVANÝCH ÚLOH

P IV - SPRÁVNÉ ODPOVĚDI K UKÁZKÁM

P V – SOUPIS PŘILOŽENÝCH SOUBORŮ

PŘÍLOHA P I: VÝPISY ŘEŠENÍ ÚLOH

Řešení úlohy mensus.net, teoretická část „Zebra puzzle“, praktická část „Zebra puzzle“

VÝPIS RELACÍ: (relace vloženy dle zadání)

REL:0 - NATIONALITY Greek | = | HOUSE 3
REL:2 - HOUSE 3 | = | COLOR pink
REL:3 - SMOKER Kools | ?B A ?B | DRINK coffee
REL:4 - PET mice | A - B | COLOR gray
REL:5 - COLOR magenta | ?B - - A - - ?B | NATIONALITY Spanish
REL:6 - PET birds | A ?B ?B... | DRINK milk
REL:7 - NATIONALITY British | ?B - - A - - ?B | SMOKER Chesterfields
REL:8 - SMOKER Pall Mall | ?B A ?B | COLOR yellow
REL:9 - NATIONALITY German | A - B | NATIONALITY Greek
REL:10 - COLOR magenta | A - B | COLOR yellow
REL:11 - PET horses | ?B - - A - - ?B | SMOKER Pall Mall
REL:12 - PET tortoises | A - B | DRINK tea
REL:13 - SMOKER Blend | ?B A ?B | COLOR magenta
REL:14 - DRINK espresso | ?B - - A - - ?B | COLOR red

VÝPIS ŘEŠITELE:

Rel: 0 (NATIONALITY Greek | = | HOUSE 3) Eliminate: 4
Vyreseno: NATIONALITY sloupec: 3 hodota: Greek
Rel: 2 (HOUSE 3 | = | COLOR pink) Eliminate: 4
Vyreseno: HOUSE sloupec: 3 hodota: 3
Vyreseno: COLOR sloupec: 3 hodota: pink
Rel: 4 (PET mice | A - B | COLOR gray) Eliminate: 5
Rel: 6 (PET birds | A ?B ?B... | DRINK milk) Eliminate: 2
Rel: 7 (NATIONALITY British | ?B - - A - - ?B | SMOKER Chesterfields) Eliminate: 1
Rel: 9 (NATIONALITY German | A - B | NATIONALITY Greek) Eliminate: 3
Vyreseno: NATIONALITY sloupec: 1 hodota: German
Vyreseno: NATIONALITY sloupec: 3 hodota: Greek
Rel: 10 (COLOR magenta | A - B | COLOR yellow) Eliminate: 6
Vyreseno: COLOR sloupec: 2 hodota: magenta
Vyreseno: COLOR sloupec: 4 hodota: yellow
Vyreseno: COLOR sloupec: 5 hodota: gray
Vyreseno: COLOR sloupec: 1 hodota: red
Vyreseno: COLOR sloupec: 4 hodota: yellow
Rel: 11 (PET horses | ?B - - A - - ?B | SMOKER Pall Mall) Eliminate: 2
Rel: 12 (PET tortoises | A - B | DRINK tea) Eliminate: 4
Rel: 13 (SMOKER Blend | ?B A ?B | COLOR magenta) Eliminate: 3
Vyreseno: COLOR sloupec: 2 hodota: magenta
Rel: 14 (DRINK espresso | ?B - - A - - ?B | COLOR red) Eliminate: 4
Vyreseno: DRINK sloupec: 4 hodota: espresso
Vyreseno: COLOR sloupec: 1 hodota: red
Rel: 3 (SMOKER Kools | ?B A ?B | DRINK coffee) Eliminate: 1
Rel: 4 (PET mice | A - B | COLOR gray) Eliminate: 1
Vyreseno: PET sloupec: 3 hodota: mice
Vyreseno: COLOR sloupec: 5 hodota: gray
Rel: 5 (COLOR magenta | ?B - - A - - ?B | NATIONALITY Spanish) Eliminate: 2
Vyreseno: COLOR sloupec: 2 hodota: magenta
Vyreseno: NATIONALITY sloupec: 5 hodota: Spanish
Rel: 7 (NATIONALITY British | ?B - - A - - ?B | SMOKER Chesterfields) Eliminate: 2
Rel: 8 (SMOKER Pall Mall | ?B A ?B | COLOR yellow) Eliminate: 3

Vyreseno: SMOKER sloupec: 5 hodota: Pall Mall
Vyreseno: SMOKER sloupec: 1 hodota: Chesterfields
Vyreseno: SMOKER sloupec: 3 hodota: Blend
Vyreseno: COLOR sloupec: 4 hodota: yellow
Rel: 11 (PET horses | ?B - - A - - ?B | SMOKER Pall Mall) Eliminate: 3
Vyreseno: PET sloupec: 2 hodota: horses
Vyreseno: PET sloupec: 1 hodota: tortoises
Vyreseno: PET sloupec: 4 hodota: birds
Vyreseno: PET sloupec: 5 hodota: butterflies
Vyreseno: SMOKER sloupec: 5 hodota: Pall Mall
Rel: 12 (PET tortoises | A - B | DRINK tea) Eliminate: 1
Vyreseno: PET sloupec: 1 hodota: tortoises
Vyreseno: DRINK sloupec: 3 hodota: tea
Rel: 3 (SMOKER Kools | ?B A ?B | DRINK coffee) Eliminate: 1
Rel: 6 (PET birds | A ?B ?B... | DRINK milk) Eliminate: 1
Vyreseno: PET sloupec: 4 hodota: birds
Vyreseno: DRINK sloupec: 5 hodota: milk
Vyreseno: DRINK sloupec: 1 hodota: coffee
Vyreseno: DRINK sloupec: 2 hodota: wine
Rel: 7 (NATIONALITY British | ?B - - A - - ?B | SMOKER Chesterfields)
Eliminate: 1
Vyreseno: NATIONALITY sloupec: 4 hodota: British
Vyreseno: NATIONALITY sloupec: 2 hodota: Swede
Vyreseno: SMOKER sloupec: 1 hodota: Chesterfields
Rel: 3 (SMOKER Kools | ?B A ?B | DRINK coffee) Eliminate: 1
Vyreseno: SMOKER sloupec: 2 hodota: Kools
Vyreseno: SMOKER sloupec: 4 hodota: Marlboro
Vyreseno: DRINK sloupec: 1 hodota: coffee
OK VYRESENO!

PŘÍLOHA P II: UKÁZKY ZDROJOVÝCH KÓDŮ KLÍČOVÝCH ČÁSTÍ PROGRAMŮ

Funkce `PermutatorArr(A,c)` a jí volaná funkce `RowIter(A,mask,l,p)` slouží k prohledání pole položek kde vyhledávají možných řešení. Prvky, které nebyly součástí žádného řešení, označí jako vyloučené (`false`).

```
function PermutatorArr(&$auxA,$cols){
    $possible=array(); //matice false, true bude znamenat mozne reseni, 2D
    for ($i=1;$i<=$cols;$i++){
        $possible[$i]=array_fill(1, $cols,false);
    }
    RowIter($auxA,array_fill(1, $cols, true),1,$possible);
    //projde matice aux a possible, provede aux AND possible, ulozi do aux
    //vrati pocet zmen - mnozstvi odstraneni nepripustnych hodnot
    //pouziva se v matici atributu k odstraneni nepripustnych kombinaci
    $cnt=0;
    $log="";
    for($i=1; $i<=$cols;$i++){//neni foreach protoze v aux je i s - reseni
        foreach ($auxA[$i] as $keyC=>$selement){
            if ($selement && !$possible[$i][$keyC]) {
                $cnt++;
                $log="!!!Vyloucena variace, radek:". $i." sloupec:". $keyC."<br
/>\n";
            }
            $selement=$selement && $possible[$i][$keyC];
        }
    }
    return array("res"=>$cnt,"log"=>$log);
}

function RowIter(&$auxA,$mask,$level,&$possible){
    $ret=false;
    foreach($auxA[$level] as $key=>$value){
        if($value && $mask[$key]){
            if($level==sizeof($mask)){ //posledni rekurze
                $possible[$level][$key]=true;
                $ret=true;
            }else{
                $cv=$mask;
                $cv[$key]=false; //zapise svou pozici v mask
                if(RowIter($auxA,$cv,$level+1,$possible)){ //rekurze na level+1
                    $possible[$level][$key]=true;
                    $ret=true;
                }
            }
        }
    }
    return $ret;
}
```

Funkce `S_permute` vrátí všechny permutace v poli položek `$items`.

```
function S_permute($items, $perms = array( )) {
    if (empty($items)) {
        $return = array($perms);
    } else {
```



```

$return = array();
for ($i = count($items) - 1; $i >= 0; --$i) {
    $newitems = $items;
    $newperms = $perms;
    list($foo) = array_splice($newitems, $i, 1);
    array_unshift($newperms, $foo);
    $return = array_merge($return, S_permute($newitems,
$newperms));
}
}
return $return;
}

```

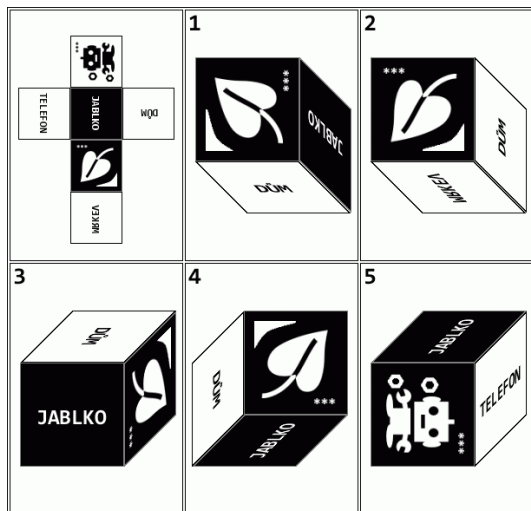
Funkce PrermutatorAtr() prohledá všechny možnosti a u každé zkontroluje platnost vazeb.

```

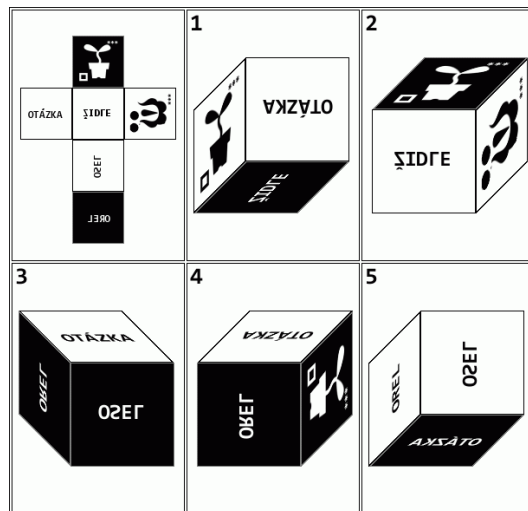
public function PrermutatorAtr(){
    $tmp=S_permute(range(1,$this->cols));
    foreach($tmp as $key=>$p_row){
        $this->solve[2]=array_combine(range(1,$this->cols), $p_row);
        foreach ($this->relation as $key =>$rel){ //prochazi vsechny
relace
            $relT=$rel["rel_type"];
            $possible=($relT>0)?false:true;
            $pA=array_search($rel["A"],$this->solve[$rel["cA"]]); //vrati
posici prvku A
            $pB=array_search($rel["B"],$this->solve[$rel["cB"]]); //vrati
posici prvku A
            foreach($this->RelMatrix($relT) as $offset){ //projde se matice
relation
                if (($pA+$offset)==$pB){
                    if ($relT>0){ // negativni pravidlo
                        $possible=true;
                        break;
                    }else{
                        $possible=false;
                        break;
                    }
                }
            }
            if($possible==false) break;
        }
        if ($possible==true) return "NALEZENO RESENI!";
    }
    return "NENALEZENO RESENI!";
}

```

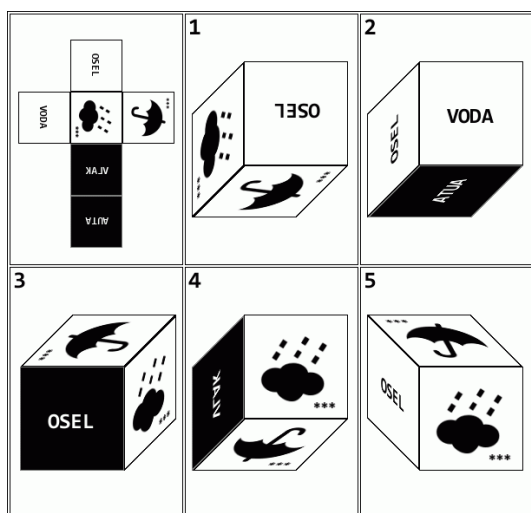
PŘÍLOHA P III: UKÁZKY GENEROVANÝCH ÚLOH



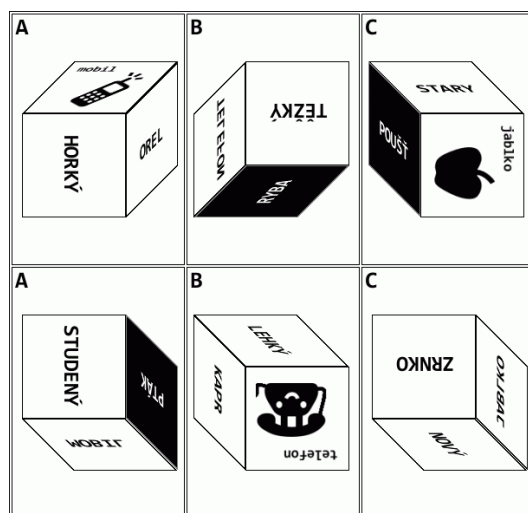
Příklad 1: Který z pohledů NEODPOVÍDÁ plánu kostky?



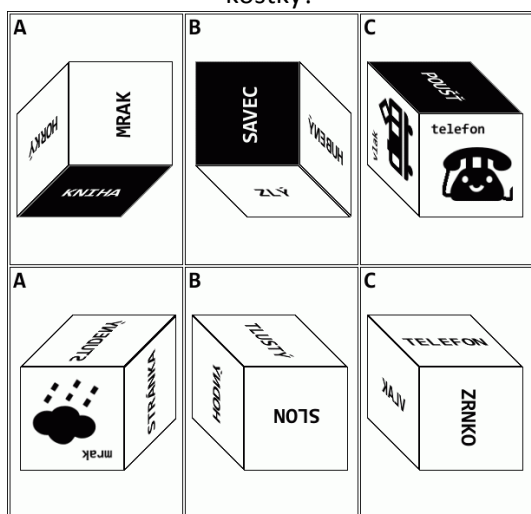
Příklad 2: Který z pohledů ODPOVÍDÁ plánu kostky?



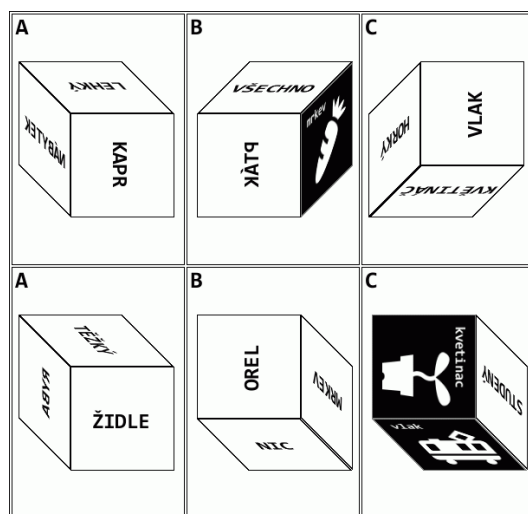
Příklad 3: Který z pohledů ODPOVÍDÁ plánu kostky?



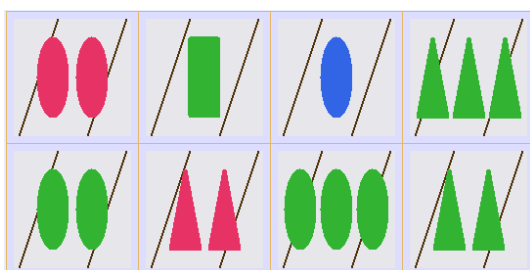
Příklad 4: Která kostka nepatří mezi ostatní?



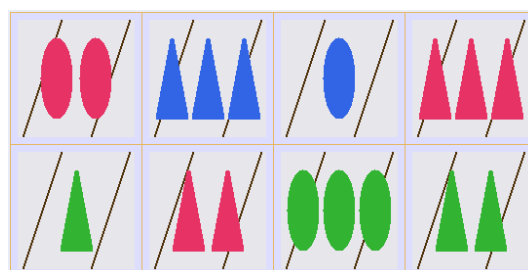
Příklad 5: Která kostka nepatří mezi ostatní?



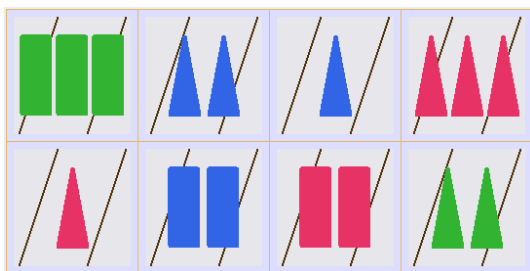
Příklad 6: Která kostka nepatří mezi ostatní?



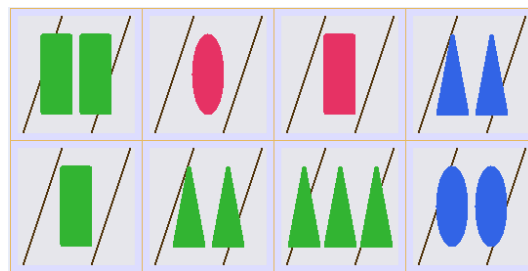
Příklad 7: Najdi SET.



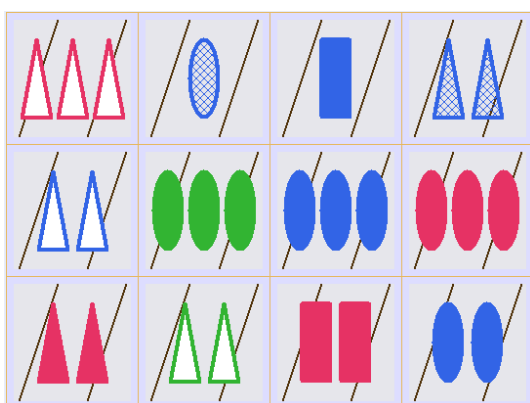
Příklad 8: Najdi SET.



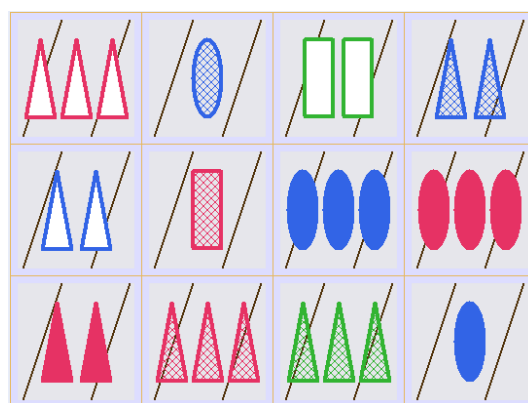
Příklad 9: Najdi SET.



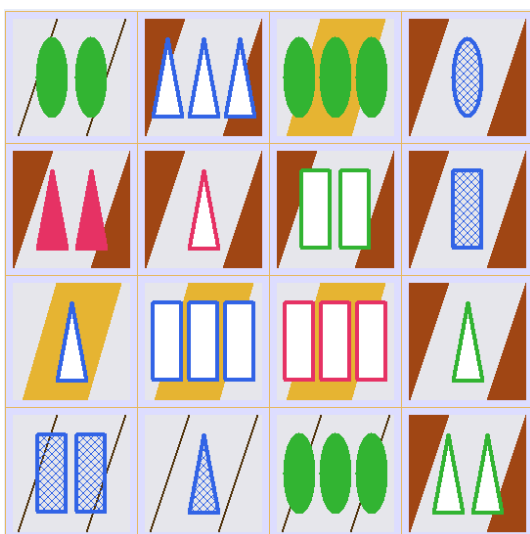
Příklad 10: Najdi SET.



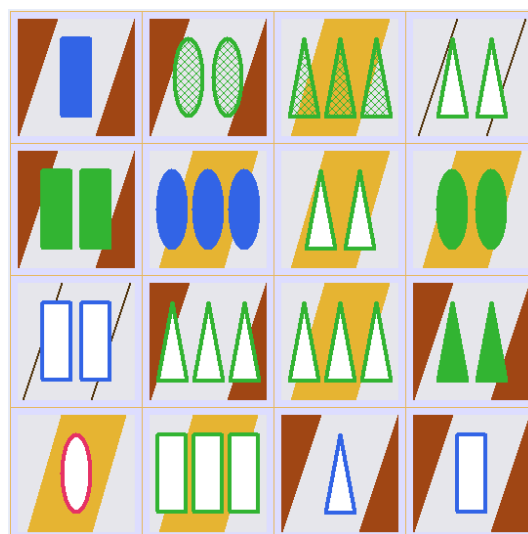
Příklad 11: Najdi SET.



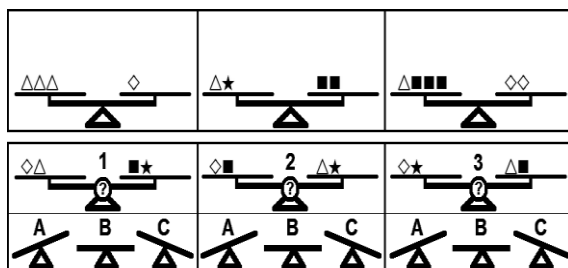
Příklad 12: Najdi SET.



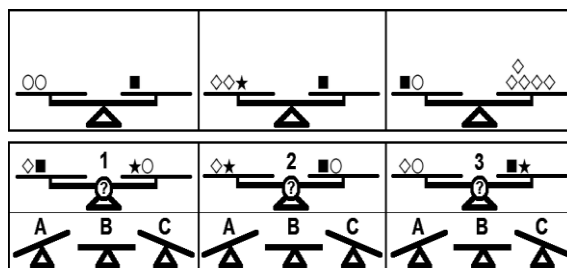
Příklad 13: Najdi SET.



Příklad 14: Najdi SET.



Příklad 15: co ukážou spodní váhy 1,2,3



Příklad 16: co ukážou spodní váhy 1,2,3

Příklad 17 – zebra puzzle:

- Kendista bydlí ob jedno vpravo od Adamova domu.
 - V druhém domě bydlí Rakušan.
 - V čtvrtém domě bydlí David.
 - David bydlí ob jedno vpravo od hokejistova domu.
 - Rakušan bydlí vpravo hned vedle zeleného domu.
 - David bydlí přímo vedle Moravákova domu.
 - V třetím domě bydlí Božetěch.
 - Božetěch bydlí ob jedno vpravo od judistova domu.
 - Judista bydlí ob jedno vedle od modrého domu.
 - Judista není Němec.
- David bydlí ob jedno vpravo od Adamova domu.
- Basketbalista nebydlí vpravo přímo vedle bílého domu.

Určete, kde kdo bydlí:





















BARVA: Zelená / Hnědá / Modrá / Bílá

JMÉNO: Cyril / Adam / Božetěch / David





































ZEMĚ: Itálie / Rakousko / Morava / Německo

SPORT: Judo / Hokej / Basketbal / Kendo

Příklad 18 – zebra puzzle:

Rozmístěte symboly pod čarou tak, aby jejich umístění splňovalo podmínky vzájemného umístění z pravého obrázku. (Inverzní obraz znamená negaci – prvek není na tomto místě).

PŘÍLOHA P IV: SPRÁVNÉ ODPOVĚDI K UKÁZKÁM

Příklad 1: pohled 5 – obrázek robota je invertován.

Příklad 2: pohled 2 –

Pohled 1 - 🦿 | ŽIDLE zamena.
Pohled 3 - OSEL inverze.
Pohled 4 - OREL | 🦿 zamena.
Pohled 5 - OTÁZKA | OREL zamena.

Příklad 3: pohled 5 –

Pohled 1 - 🦿 rotace.
Pohled 2 - AUTA flip.
Pohled 3 - OSEL inverze.
Pohled 4 - 🦿 rotace.


Příklad 4: kostka A – strana „HORKÝ“ je otočena nesprávně

Příklad 5: kostka C – strana „VLAK“ je neprávne stranově otočena

Příklad 6: kostka C- strany „KVĚTINÁČ“ a „HORKÝ“ jsou mezi sebou vyměněny

Příklad 7:  3 řešení

Příklad 8:  2 řešení

Příklad 9:  1 řešení

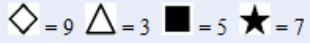
Příklad 10:  1 řešení

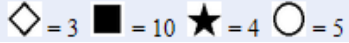
Příklad 11:  3 řešení

Příklad 12:  1 řešení

Příklad 13:  2 řešení

Příklad 14:  1 řešení





















Příklad 15: B(12=12), A(14>10), A(16>8) 

Příklad 16: A(13>9), C(7<15), C(8<14) 

Příklad 17: Řešení

1	2	3	4
Hnědá	Modrá	Zelená	Bílá
Božetěch	Cyril	David	Adam
Rakousko	Morava	Německo	Itálie
Hokej	Judo	Kendo	Basketbal

Příklad 18: Řešení

PŘÍLOHA P V: POPIS PŘILOŽENÝCH SOUBORŮ

```
z_array.php //pomocny soubor pro zebra.php
z_balance.php //vahy
z_card.php //SET
z_dice.php //kostka
z_function.php //pomocne funkce
z_invest.php //invest
z_math.php //pomocny soubor pro z_balance.php
z_score.php //soubor pro praci se score pristupny z her i hosting.
portalu
z_sudra.php //SUDRA
//dale provadi log do databaze
zebra.php //zebra
zscore.php //okenni zobrazeni score v hostujicim portalu s detaily
zscore_view.php //banner do hostujiciho portalu

z-lab\ //adresar pro sobory her a pomocne
\z-lab\game.sql //struktura tabulky game (pro log her)
\z-lab\invest.sql //struktura tabulky invest (nutna pro z_invest)
\z-lab\filuta_gen.py //generator obrazku pro SET
\z-lab\filuta.py //puvodni hra SET v pythnou

\z-lab\z_balance
\z-lab\z_card
\z-lab\z_dice
\z-lab\z_invest
\z-lab\z_sudra
\z-lab\zebra

//kazdy obsahuje:
icon.ico //ikona hry
script.js //sobor se scripty
style.css //soubor s kaskadovym stylem

//kazda hra musi obsahovat:
session_start(); //start session
require_once("z_function.php"); //pripojuje k databazi a autoload classes
require_once("z_score.php"); //pristup ke score
define ("PATH","z-lab/zebra/"); //definuje umisteni ico,css,js,img a
pomocnych

//z_function.php pripojuje k databazi (potrebuje z_score a z_invest)
//parametry jsou v def.php ktery zavadi a ktery musi obsahovat:
define ("DB_USER","****"); // uzivatel
define ("DB_PASS","****"); // heslo
define ("DB_NAME","****"); // jmeno databaze

\classes\ScriptVar.php //v \classes jsou hledany classes pomoci autoload
//tato trida zapisuje do html var pro script
```