Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Softvérová podpora vyučovania diskrétnej matematiky

Diplomová práca

2021

Maroš Malý

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Softvérová podpora vyučovania diskrétnej matematiky

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	Aplikovaná informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	Ing. Ján Komara, PhD.

Bratislava, 2021

Maroš Malý

76818218

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Maroš Malý
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Softvérová podpora vyučovania diskrétnej matematiky
Educational software for discrete mathematics

Anotácia: Návrh a implementácia abstraktných dátových typov pre tieto konečné matematické štruktúry: binárne relácie, relácie na množine, čiastočné usporiadania a zobrazenia. Ich základná množinová reprezentácia je rozšírená o grafový a maticový pohľad. Návrh a implementácia jazyka dopytov pre testovanie jednoduchých vlastností takýchto štruktúr. Ako implementačný jazyk je zvolený programovací jazyk Python v prostredí Jupyter Notebook. Výsledná implementácia je publikovaná v niektorej zo slobodných licencií kompatibilnej s GNU GPL.

Cieľ: Cieľom práce je navrhnúť, implementovať a otestovať interaktívne nástroje v prostredí Jupyter Notebook pre podporu výučby úvodného kurzu diskrétnej matematiky.

Literatúra: Discrete and combinatorial mathematics : An applied introduction / Ralph P. Grimaldi. Boston : Pearson/Addison-Wesley, 2004

Data Structures and Algorithms in Python / Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser. Wiley, 2013.

Learning IPython for Interactive Computing and Data Visualization / Cyrille Rossant. Packt Publishing, 2nd edition, 2015.

Vedúci: Ing. Ján Komara, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 18.11.2020

Dátum schválenia: 19.11.2020 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

....
študent vedúci práce

Poďakovanie: ...

Abstrakt

...

Kľúčové slová: ...

Abstract

...

Keywords: ...

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Diskrétna matematika je odbor, ktorý sa zaoberá diskrétnymi matematickými štruktúrami. Štruktúrami, pre ktoré môžeme charakterizovať pomocou celých čísel a sú teda počítateľné.

Rozvoj diskrétnej matematiky podmienil rozvoj aj samotnej informatiky. Diskrétna matematika sa často chápe ako časť informatiky a zvyknú sa do nej zaraďovať práve nové matematické disciplíny, ktoré vznikli v súvislosti s rozvojom výpočtovej techniky.

Matematické štruktúry, ktorými sa zaoberá diskrétna matematika, sú charakteristické tým, že aj najmenšia zmena vstupných hodnôt alebo prvkov vie mať za následok obrovskú zmenu výstupných hodnôt alebo celku.

Významným konceptom využívaným v diskrétnej matematike sú množiny. Koncepcia množiny patrí medzi základné formálne prostriedky matematiky. Umožňuje formulovať prehľadným a jednotným spôsobom všetky oblasti matematiky prostredníctvom elementárnej štruktúry množiny a operáciami nad ňou. Teória množín vznikla koncom 19. storočia hlavne zásluhou nemeckého matematika Georga Cantora (1845 – 1918). Zásluhu na jej rozšírení má anglický logik a filozof Bertrand Russell (1872 – 1970), ktorý objavil vnútorné nekonzistentnosti jej intuitívnej formulácie, kt oré boli neskôr prekonané jej dôslednou axiomatickou výstavbou.

V mnohých aplikáciách množinová dátová štruktúra podstatne uľahčuje implementáciu algoritmov, ktoré sú založené na formalizme teórie množín. Ako príklad takýchto algoritmov môže slúžiť teória grafov, ktorej jednoduchá a súčasne aj elegantná teória je založená na množinách. Mnohé algoritmy teórie grafov (napr. problém obchodného cestujúceho) patrí medzi základné algoritmy, preto je dôležité, hlavne z pedagogických dôvodov, mať možnosť využívať dátovú štruktúru množiny pre zjednodušenie a sprehľadnenie týchto algoritmov.

Nakoľko je pochopenie a osvojenie si týchto konceptov pre študentov kritické, tak je aj potreba vytvorenia softvérovej podpory, ktorá im celý tento proces spríjemní a zjednoduší. Študenti si môžu vďaka softvéru precvičovať osvojené vedomostia a zároveň a testovať a experimentovať s predpripravenými diskrétnymi matematickými štruktúrami. Vytvorený nástroj študentom poskytuje interaktívne prostredie Jupyter Notebook, kde si využitím pre nich pripraveného jazyka dopytov pre dané štruktúry, môžu testovať jednoduché vlastnosti týchto štruktúr využitím programovacieho jazyka Python.

1 Východiská

1.1 Technologické východiská

V tejto kapitole si priblížime technologické východiská diplomovej práce.

1.1.1 Python

Python je interpretovaný, interaktívny programovací jazyk, ktorý vytvoril Guido van Rossum, pôvodne ako skriptovací jazyk pre Amoeba OS schopný systémových volaní. Python je vyvíjaný ako open source projekt.

Python je multi-paradigmový jazyk podobne ako Perl. To znamená, že namiesto toho aby nútil programátora používať určitý štýl programovania, umožňuje používanie viacerých. Python podporuje objektovo orientované, štruktúrované aj funkcionálne programovanie. Je to dynamicky typový jazyk, podporuje veľké množstvo vysokoúrovňových dátových typov a na správu pamäte používa garbage collection.

Aj keď bol Python pôvodne vyvíjaný ako „skriptovací jazyk", používa sa na vývoj mnohých veľkých softvérových projektov ako sú aplikačný server Zope a systémy na zdieľanie súborov Mnet a BitTorrent. Tak isto ho široko využíva Google. Zástancovia Pythonu ho radšej volajú vysokoúrovňovým dynamickým programovacím jazykom, lebo pojem „skriptovací jazyk" sa asociuje s jazykmi, ktoré sa používajú len na jednoduché shell skripty alebo s jazykmi ako JavaScript: jednoduchšími a na väčšinu účelov menej spôsobilými ako „skutočné" programovacie jazyky ako Python.

Python sa dá jednoducho rozširovať. Nové zabudované moduly môžu byť jednoducho napísané v C alebo C++. Python tiež môže byť použitý ako rozširovací jazyk pre existujúce moduly a aplikácie, ktoré potrebujú programovateľné rozhranie.

Aj keď návrhár Pythonu je trochu nepriateľský k funkcionálnemu programovaniu a k tradícii Lispu, sú tu viditeľné paralely medzi filozofiou Pythonu a filozofiou minimalistických jazykov Lispovej rodiny ako sú Scheme. Kvôli tomu veľa bývalých programátorov v Lispe považuje Python za príťažlivý. [4][11]

1.1.2 Jupyter

Projekt Jupyter je označenie pre skupinu softvérových produktov zprístupňujících programovanie pomocou webového rozhrania a vyvíjaných pod hlavičkou neziskovej organizácie Project Jupyter. Názov odkazuje jednak na tri hlavné podporované programovacie jazyky, kterými sú Julia, Python a R, jednak na vedecký zápisník Sidereus nuncius, do ktorého si Galileo Galilei zapisoval svoje objevovania mesiacov Jupitera.

Projekt založil Fernando Pérez v roku 2014, pričom nadviazal na čisto pythonovský projekt IPython, ktorého jazykovo závislé súčasti sú aj naďalej vyvíjené a sú súčasťou projektu Jupyter, zatiaľčo jazykovo menej závislé a nezávislé súčasti sa vyvíjajú v rámci projektu Jupyter zvlášť a podporujú už aj dalšie jazyky. Postupne tak bola do Jupytera pridaná podpora pre Juliu, R, Haskell, Ruby, Clojure, Go a mnohé dalšie.

Hlavnou a nejznámejšou súčasťou projektu Jupyter je Jupyter Notebook, bohatá webová aplikácia pre vytváranie dokumentov, které okrem zdrojového kódu môžu obsahovat aj grafy, tabuľky, matematické vzorce (podporované je ich vkladanie v syntaxi TeXu a LaTeXu), diagramy UML, rastrové obrázky a iné.

V roku 2018 získal projekt cenu ACM Software System Award za rok 2017. [12][13]

1.1.3 Jupyter Notebook

Jedná sa o interaktívne výpočtové prostredie, ktoré umožňuje používateľom experimentovať s kódom a zdieľať ho.

Jupyter je skratka pre Juliu, Pythona a R., tri programovacie jazyky, s ktorými Jupyter začínal, aj keď dnes podporuje veľké množstvo jazykov.

Je široko používaný na vytváranie a zdieľanie dokumentov obsahujúcich kód. To je pri výučbe veľmi užitočné, pretože na príkladoch môžeme ukázať, ako funguje skript, jazyk alebo požiadať študentov, aby navrhli a overili svoj vlastný kód. [12]

1.1.4 Graphviz

Graphviz je balík slobodného softéru pre kreslenie grafov zadaných vo formáte DOT. Jeho vývoj započal v AT&T, v súčasnosti se na ňom podieľajú dobrovoľníci a Graphviz je k dispozícii pod licenciou Common Public License. [1]

1.1.5 SAT solver

SAT solver je počítačový program využívaný v informatike ale aj v iných odvetviach, ktorého cieľom je riešiť problém splniteľnosti booleanovského výrazu. Na vstup dostáva SAT solver formulu nad boolean premennými, ako napríklad "(x or y) and (x or not y)" resp. konjunkciu disjunkcií. Výstupom SAT solvera je informácia či je daná formula na vstupe splniteľná, čiže či existuje také ohodnotenie premenných x a y, pre ktorú je výsledná formula pravdivá, prípadne nesplniteľná a teda neexistuje také ohodnotenie x a y, pre ktoré by bola formula pravdivá. Prvé SAT algoritmy datujeme už od 60-tych rokov. Dnešné moderné SAT solvery sú veľmi komplexné programy, obsahujúce veľké množstvo heuristík a optimalizácií aby dokázali pracovať čo najefektívnejšie a na čo najväčšie formuly.

Nakoľko problém splniteľnosti je NP úplný problém algoritmy pre jeho riešenie existujú iba exponenciálnej zložitosti. Napriek tomu SAT solveri a hlavne ich rozvoj začiatkom druhého milénia výrazným spôsobom prispeli k obrovskému pokroku v našej dnešnej schopnosti automaticky riešiť problémy obsahujúce desiatky tisíc premenných a milióny obmedzení.

Zvyčajne SAT solver prevedie najprv formulu do konjuktívnej normálovej formy. Ich základný algoritmus je založený na algoritmoch ako DPLL s tým, že ako nadstavbu má rôzne rozšírenia a vlastnosti. Väčšina SAT solverov obsahuje časovač a po uplynutí rozumne dlhého času výpočet zastavia ak sa im nepodarí riešenie (ak vôbec existovalo) nájsť. V takom prípade vyhlási, že výsledok splniteľnosti danej formuly je neznámy. Pre splniteľné formuly SAT solvery zvyknú okrem informácie či je formula splniteľná poskytnúť aj konkrétne ohodnoteie formuly, pre ktoré je formula splniteľná.

Moderné SAT solvery majú významný vplyv na oblasti ako napríklad verifikácia softvéru, analýza programov, kombinatorické problémy, umelá inteligencia, elektornická automatizácia návrhu a iných. [8]

1.1.6 MiniSat

MiniSat je minimalistický, open-source SAT solver, vyvinutý pre výskumníkov ako aj vývojárov pre jednoduchšie začatie používania SAT solverov. Je vydaný pod MIT licenciou a momentálne sa využíva v mnohých projektoch. Spolu s SatELite vyhral MiniSat v roku 2005 jednu z kategórii v súťaži SAT solverov.

Hlavné vlastnosti Minisatu:

(1) Jednoduchý na upravovanie. MiniSat je malý, jednoduchý a s výbornou a poriadnou dokumentáciou a teda je aj ideálnym odrazovým mostíkom pre prácu so SAT solvermi.

(2) Vysoko efektívny. MiniSat je dobrý začiatočný bod pre budúci výskum v SAT a pre aplikácie využívajúce SAT.

(3) Navrhovaný pre integráciu. MiniSat podporuje incremental SAT a má mechanizmy pre pridávanie neklauzálnych podmienok. Vďaka jednoduchosti upravovania MiniSatu je dobrou voľbou ako backend iného nástroja, zaoberajúceho sa inou špecifickejšou problematikou. [3]

1.2 Teoretické východiská

V tejto kapitole si priblížime teoretické východiská diplomovej práce.

1.2.1 Množiny

Množina je súbor neusporiadaných prvkov. Ak sa nejaké dva prvky nachádzajú v tej istej množine, potom musia byť od seba odlíšiteľné.

Nech A je množina, potom ak prvok x patrí (nepatrí) do množiny A označíme výrazom $x \in A$ ($x \notin A$). [7]

Špecifikácia množiny. Množinu vieme špecifikovať dvoma spôsobmi.

(1) Vymenovaním všetkým prvkov, ktoré do nej patria

$$A = \{x_1, x_2, x_3, ..., x_n\}$$

(2) Využitím predikátu $P(x)$, ktorý určuje, či prvok $x \in U$ (zo zvoleného univerza U) patrí do množiny A alebo nepatrí.

$$A = \{x \in U; P(x)\}$$

[7]

Charakteristická funkcia. Pomocou predikátu $P(x)$ zadefinujeme charakteristickú funkciu μ_A:

$$P(x) =_{def} (\mu_A(x) = 1)$$

kde

$$\mu_A(x) = \begin{cases} 1 & \text{x} \in \text{A} \\ 0 & \text{x} \notin \text{A} \end{cases}$$

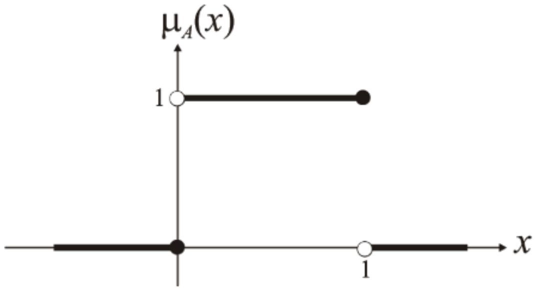
Množina A nad univerzom U je definovaná pomocou charakteristickej funkcie nasledovne:

$$A = \{x \in U; \mu_A(x) = 1\}$$

Kde charakteristická funkcia μ_A je zobrazenie

$$\mu_A : U \rightarrow \{0, 1\}$$

ktoré binárne ohodnotí každý prvok x univerza U - 1 (0) ak prvok x patrí (nepatrí) do množiny A. [7]



Obr. 1.1: Charakteristická funkcia množiny A, ktorá je určená ako polootvorený interval $A = (0, 1 >$

Rovnosť množín. Množina $A = \{x; \mu_A(x) = 1\}$ sa rovná množine $B = \{x; \mu_B(x) = 1\}$, $A = B$, vtedy a len vtedy, ak sú obe definované nad rovnakým univerzom U a charakteristické funkcie oboch množín sú rovnaké.

$$(A = B) =_{def} \forall(x \in U)(\mu_A(x) = \mu_B(x))$$

[7]

Podmnožina. Množina $A = \{x; \mu_A(x) = 1\}$ je podmnožinou množiny $B = \{x; \mu_B(x) = 1\}$, $A \subseteq B$, vtedy a len vtedy, ak každý prvok z množiny A patrí do množiny B

$$(A \subseteq B) =_{def} \forall(x \in U)((\mu_A(x) = 1) \Rightarrow (\mu_B(x) = 1))$$

[7]

Vlastná podmnožina. Množina $A = \{x; \mu_A(x) = 1\}$ je vlastnou podmnožinou množiny $B = \{x; \mu_B(x) = 1\}$, $A \subset B$, vtedy a len vtedy, ak

$$(A \subset B) =_{def} (A \subseteq B \wedge A \neq B)$$

[7]

Zjednotenie množín. Množina $A \cup B$ je zjednotenie množín A a B, vtedy a len vtedy, ak

$$(A \cup B) =_{def} \{x; (x \in A) \vee (x \in B)\} = \{x; \mu_{A\cup B}(x) = 1\}$$

kde

$$\mu_{A\cup B}(x) = max\{\mu_A(x), \mu_B(x)\}$$

[7]

Prienik množín. Množina $A \cap B$ je prienik množín A a B, vtedy a len vtedy, ak

$$(A \cap B) =_{def} \{x; (x \in A) \vee (x \in B)\} = \{x; \mu_{A \cap B}(x) = 1\}$$

kde

$$\mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\}$$

[7]

Komplement množiny. Množina $\overline{A}$ je komplement množiny A, vtedy a len vtedy, ak

$$\overline{A} =_{def} \{x; (x \notin A)\} = \{x; \mu_{\overline{A}}(x) = 1\}$$

kde

$$\mu_{\overline{A}}(x) = 1 - \mu_A(x)$$

[7]

Rozdiel množín. Množina $A - B$ je rozdiel množín A a B, vtedy a len vtedy, ak

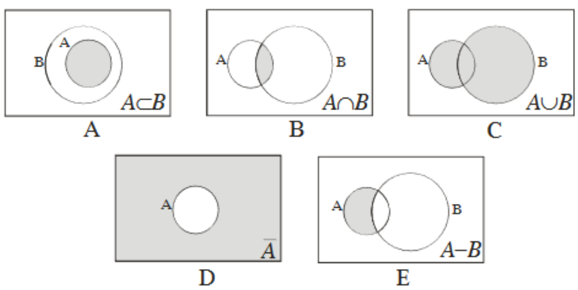
$$(A - B) =_{def} \{x; (x \in A) \wedge (x \notin B)\} = \{x; \mu_{A-B}(x) = 1\}$$

kde

$$\mu_{A-B}(x) = min\{\mu_A(x), 1 - \mu_B(x)\}$$

[7]

Obr. 1.2: Reprezentácia definovaných operácii nad množinou U pomocou Vennových diagramov



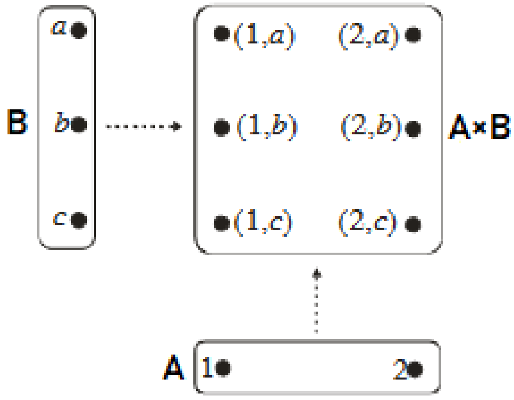
Na týchto diagramoch obdĺžniková oblasť znázorňuje univerzum U, kde množiny A a B sú podmnožiny univerza. Diagram A znázorňuje reláciu „podmnožina“ $A \subset B$, keď množina – oblasť A celá leží v množine – oblasti B. Diagram B znázorňuje operáciu „prienik“ $A\ capB$, kde vyšrafovaná oblasť reprezentuje prienik množín A a B. Porovnaním diagramov A a B zistíme, že ak $A \subset B$, potom $A \cap B = A$. Diagram C znázorňuje operáciu zjednotenia množín A a B, vyšrafovaná je oblasť, ktorá sa nachádza v množine A alebo v množine B. Diagram D znázorňuje operáciu doplnok množiny A vzhľadom k univerzu U. Diagram E znázorňuje rozdiel množín A a B, vyšrafovaná je oblasť, ktorá sa nachádza v A a súčasne sa nenachádza v B.

Karteziánsky súčin množín. Množina $A \times B$ je karteziánsky súčin množín A a B, vtedy a len vtedy, ak

$$A \times B = \{(x,y); (x \in A) \wedge (x \in B)\}$$

[7]

Obr. 1.3: Znázornenie karteziánskeho súčinu pomocou Vennových diagramov



1.2.2 Binárne relácie

Množina R je binárna relácia z množiny A do množiny B, vtedy a len vtedy, ak je podmnožinou karteziánskeho súčinu množín A a B.

$$R \subseteq A \times B$$

$$R = \{(x,y); \mu_R(x,y) = 1\}$$

[7]

Inverzná relácia. Nech $R \subseteq A \times B$ je binárna relácia, potom množina usporiadaných dvojíc $(y,x) \in B \times A$, pre ktoré platí, že $(x,y) \in A$ je inverzná relácia R^{-1} (vzhľadom k relácii R) vtedy a len vtedy, ak

$$R^{-1} = \{(y,x); (x,y) \in R\}$$

Pre binárne relácie, ktoré sú definované nad rovnakou dvojicou množín A a B vieme definovať štandardné množinové operácie.

Nech R a S sú binárne relácie definované nad rovnakou dvojicou množín A a B

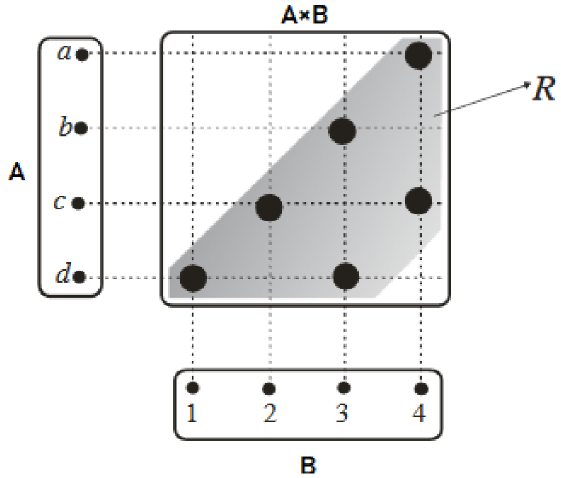
$$R,S \subseteq A \times B$$

$$R = \{(x,y); \mu_R(x,y) = 1\}$$

$$S = \{(x,y); \mu_S(x,y) = 1\}$$

[7]

Obr. 1.4: Znázornenie relácie R ako podmnožiny karteziánskeho súčinu (vytieňovaná oblasť) dvoch množín A a B, $R = \{(d,1),(c,2),(b,3),(d,3),(a,4),(c,4)\}$



Zjednotenie relácii. Binárna relácia $R \cup S$ je zjednotenie relácii R a S vtedy a len vtedy, ak

$$R \cup S = \{(x,y); \mu_{R \cup S}(x,y) = 1\}$$

$$\mu_{R \cup S}(x,y) = max\{\mu_R(x,y); \mu_S(x,y)\}$$

[7]

Prienik relácii. Binárna relácia $R \cap S$ je prienik relácii R a S vtedy a len vtedy, ak

$$R \cap S = \{(x,y); \mu_{R \cap S}(x,y) = 1\}$$

$$\mu_{R \cap S}(x,y) = min\{\mu_R(x,y); \mu_S(x,y)\} \text{ [7]}$$

Diagramy A a B znázorňujú relácie R a S definované nad rovnakými množinami A a B. Diagramy C a D znázorňujú zjednotenie resp. prienik týchto dvoch relácií. Diagramy E a F znázorňujú inverzné relácie R^{-1} resp. $S-1$. Diagramy G a H znázorňujú doplnky k reláciám R resp. S.

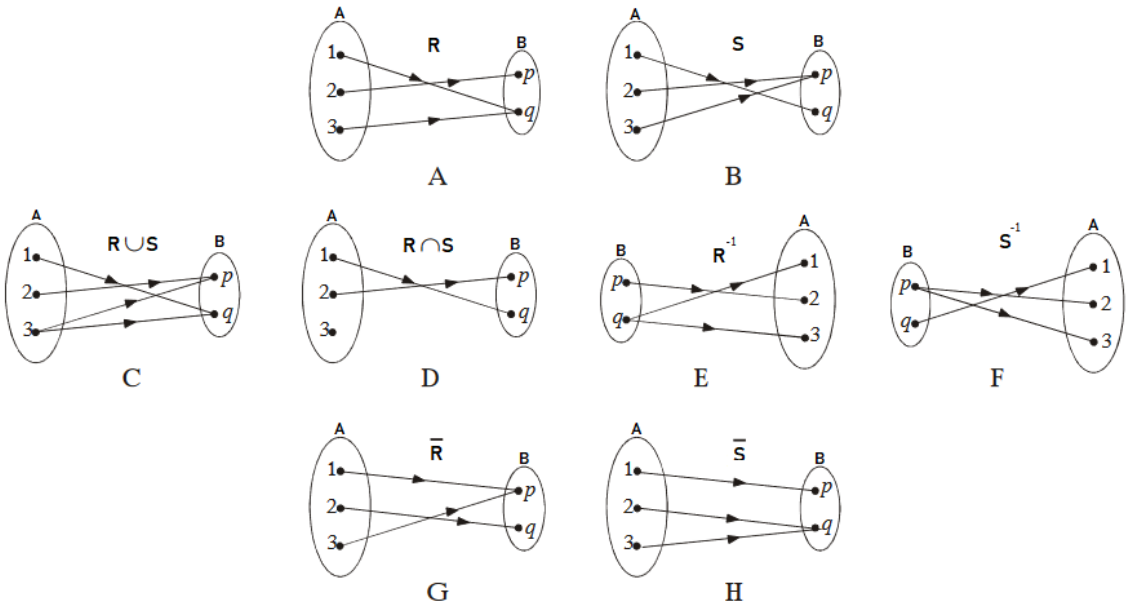
Komplement relácie. Binárna relácia $\overline{R}$ je komplement relácie R vtedy a len vtedy, ak

$$\overline{R} = \{(x,y); \mu_{\overline{R}}(x,y) = 1\}$$

$$\mu_{\overline{R}}(x,y) = 1 - \mu_R(x,y)$$

[7]

Obr. 1.5: Reprezentácia definovaných operácii nad reláciami R a S pomocou Vennových diagramov



Matica relácie. Binárna matica M je matica relácie R vtedy a len vtedy, ak pre každý maticový prvok M_{ij} platí

$$M_{ij} = \mu_R(x_i, y_j) = \begin{cases} 1 & ((x_i,y_j) \in \mathrm{R}) \\ 0 & ((x_i,y_j) \notin \mathrm{R}) \end{cases}$$

[7]

Kompozícia relácii. Nech $R = \{(x,y); \mu_R(x,y) = 1\} \subseteq A \times B$ a $S = \{(y,z); \mu_S(y,z) = 1\} \subseteq B \times C$ sú binárne relácie, potom $R \circ S = \{(x,z); \mu_R(x,z) = 1\}$ je kompozícia relácii R a S vtedy a len vtedy, ak

$$\mu_{R \circ S}(x,z) = \max_y \{min\{\mu_R(x,y), \mu_S(y,z)\}\}$$

[7]

1.2.3 Relácie na množine

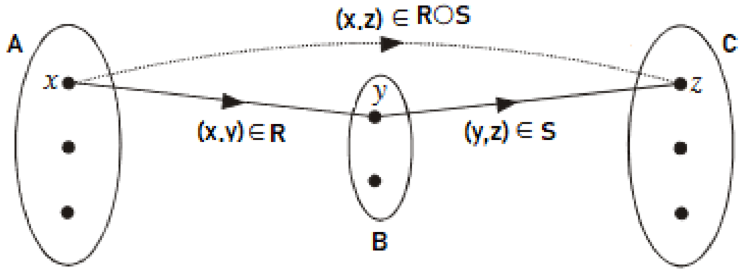
Binárna relácia R je relácia na množine A vtedy a len vtedy, ak je podmnožinou karteziánskeho súčinu množín A a A.

$$R \subseteq A \times A$$

$$R = \{(x,y); \mu_R(x,y) = 1\}$$

[7]

Obr. 1.6: Znázornenie kompozície dvoch relácií R a S, výsledná relácia $R \circ S$ obsahuje dvojicu (x,z) vtedy a len vtedy, ak existuje taký prvok $y \in B$, že platí $(x,y) \in R$ a $(y,z) \in S$.



Reflexívne relácie. Relácia na množine $R \subseteq A \times A$ je reflexívna vtedy a len vtedy, ak

$$\forall (x \in A)((x,x) \in R)$$

[7]

Symetrické relácie. Relácia na množine $R \subseteq A \times A$ je symetrická vtedy a len vtedy, ak

$$\forall (x,y \in A)((x,y) \in R \Rightarrow (y,x) \in R)$$

[7]

Antisymetrické relácie. Relácia na množine $R \subseteq A \times A$ je antisymetrická vtedy a len vtedy, ak

$$\forall (x,y \in A)((x,y) \in R \wedge (y,x) \in R \Rightarrow x = y)$$

[7]

Asymetrické relácie. Relácia na množine $R \subseteq A \times A$ je asymetrická vtedy a len vtedy, ak

$$\forall (x,y \in A)((x,y) \in R \Rightarrow (y,x) \notin R)$$

[7]

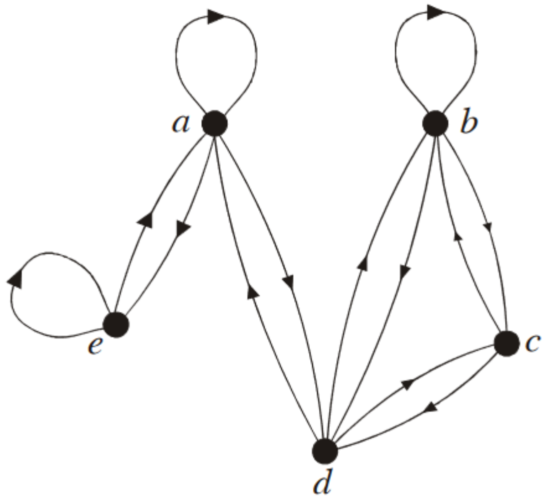
Tranzitívne relácie. Relácia na množine $R \subseteq A \times A$ je tranzitívna vtedy a len vtedy, ak

$$\forall (x,y,z \in A)((x,y) \in R \wedge (y,z) \in R \Rightarrow (x,z) \in R)$$

[7]

Z obrázku vidíme, že (1) relácia R nie je reflexívna, potrebné slučky neexistujú na vrcholoch c a d, (2) relácia R je symetrická, ak medzi vrcholmi x a y existuje hrana (x,y), potom existuje aj opačná hrana (y,x), (3) relácia R nie je antisymetrická, z existencie dvojíc opačne orientovaných hrán na rôznych vrcholoch nevyplýva rovnosť týchto dvoch vrcholov. (4) relácia R nie je tranzitívna, pretože existencia hrán (e,a) a (a,d) neimplikuje existenciu hrany (e,d).

Obr. 1.7: Grafická interpretácia relácie R nad množinou $A = \{a, b, c, d, e\}$.



Relácia ekvivalencie. Relácia na množine $R \subseteq A \times A$ je reláciou ekvivalencie vtedy a len vtedy, ak je reflexívna, symetrická a tranzitívna. [7]

1.2.4 Čiastočné usporiadania

Relácia na množine $R \subseteq A \times A$ je čiastočné usporiadanie vtedy a len vtedy, ak je reflexívna, antisymetrická a tranzitívna. [7]

Najväčší prvok. Nech R je čiastočné usporiadanie množiny A. Prvok $x \in A$ je najväčší prvok množiny A, vtedy a len vtedy, ak

$$\forall y \in A (y, x) \in R$$

[2]

Najmenší prvok. Nech R je čiastočné usporiadanie množiny A. Prvok $x \in A$ je najmenší prvok množiny A, vtedy a len vtedy, ak

$$\forall y \in A (x, y) \in R$$

[2]

Maximálny prvok. Nech R je čiastočné usporiadanie množiny A. Prvok $x \in A$ je maximálny prvok množiny A, vtedy a len vtedy, ak

$$\forall y \in A ((x, y) \in R \rightarrow y = x)$$

[2]

Minimálny prvok. Nech R je čiastočné usporiadanie množiny A. Prvok $x \in A$ je minimálny prvok množiny A, vtedy a len vtedy, ak

$$\forall y \in A((y,x) \in R \rightarrow y = x)$$

[2]

Supremum. Nech R je čiastočné usporiadanie množiny A a nech $B \subseteq A$. Prvok $x \in A$ je supremum množiny B (sup B), vtedy a len vtedy, ak

$$\forall y \in B(y,x) \in R$$

$$\forall y \in A(\forall z \in B(z,y) \in R \rightarrow (x,y) \in R)$$

Infimum. Nech R je čiastočné usporiadanie množiny A a nech $B \subseteq A$. Prvok $x \in A$ je infimum množiny B (inf B), vtedy a len vtedy, ak

$$\forall y \in B(x,y) \in R$$

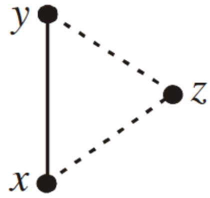
$$\forall y \in A(\forall z \in B(y,z) \in R \rightarrow (y,x) \in R)$$

[2]

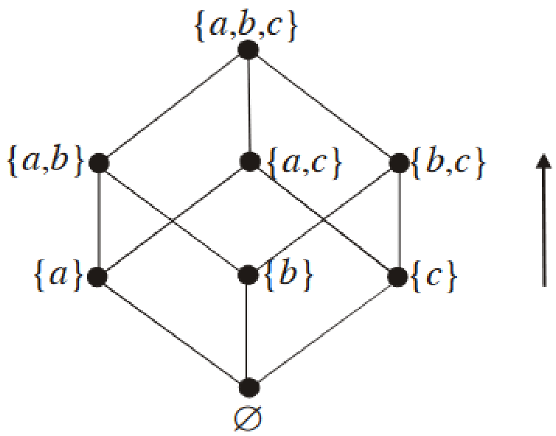
Zväz. Čiastočné usporiadanú množinu (A,R), ak pre každé dva prvky $x,y \in A$ existuje infx,y a supx,y. [2]

Hasseho diagram. Nech A je čiastočne usporiadaná množina s reláciou $R \subseteq A \times A$. Prvok y je pokrytý prvkom x vtedy, ak $(x,y) \in R$ a neexistuje taký prvok z, rôzny od x a y, pre ktorý súčasne platí $(x,z) \in R$ a $(z,y) \in R$. [7]

Obr. 1.8: Znázornenie pojmu „prvok y je pokrytý prvkom x“ , neexistuje taký prvok z, pre ktorý by platilo „prvok z je pokrytý x“ a „prvok y je pokrytý z“.



Hasseho diagram priradený konečnej množine A s reláciou $R \subseteq A \times A$ čiastočného usporiadania obsahuje vrcholy, ktoré sú stotožnené s prvkami z A, pričom dva vrcholy sú spojené hranou, ak prvok y je pokrytý prvkom x.[7]



Obr. 1.9: Znázornenie Hasseho diagramu pre potenčnú množinu množiny $A = \{a, b, c\}$, ktorá je čiastočne usporiadaná reláciou '$\subseteq$'. Šípka vpravo znázorňuje orientáciu čiar, ktoré sú orientované zdola nahor.

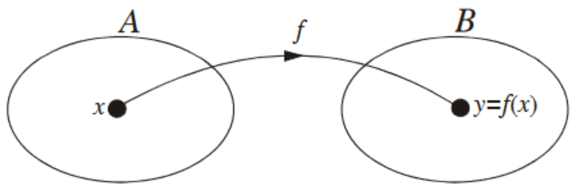
1.2.5 Zobrazenia.

Zobrazenia (funkcie) patria medzi základné pojmy matematiky. V matematike pod zobrazením f rozumieme jednoznačný predpis, pomocou ktorého každému argumentu x z množiny A priradíme práve jednu funkčnú hodnotu $y = f(x)$ z množiny B.

$$f : A \to B$$

[7]

Obr. 1.10: Schematické znázornenie zobrazenia:$f : A \to B$.

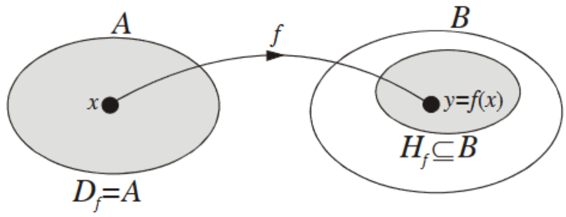


Zobrazenie je teda množinou usporiadaných dvojíc.

$$f = \{(x, f(x)); x \in A\}$$

Binárna relácia $f \subseteq A \times B$ je zobrazenie vtedy a len vtedy, ak pre každé $x \in A$ existuje práve jedno $y \in B$ také, že $(x, y) \in f$. Množina A je obor zobrazenia f, $D_f = A$, a množina B je koobor zobrazenia f.

Obor funkčných hodnôt zobrazenia f je množina $H_f = \{f(x); x \in A\} = f(A)$. Ak $(x, y) \in f$, potom x je argument a y funkčná hodnota (obraz). [7][6]



Obr. 1.11: Znázornenie skutočnosti, že pre zobrazenie $f : A \to B$, obor funkčných hodnôt $H_f = f(A)$ je vo všeobecnosti len podmnožinou B.

Rovnosť zobrazení. Zobrazenie $f : A \to B$ sa rovná zobrazeniu $g : A' \to B'$ vtedy a len vtedy, ak súčasne platí

$$A = A'$$

$$\forall(x \in A)(f(x) = g(x))$$

[7]

Jednotková funkcia. Zobrazenie $i_A : A \to A$ je jednotková funkcia vtedy a len vtedy, ak

$$\forall(x \in A)(i_A(x) = x)$$

jej obor a obor funkčných hodnôt sa rovnajú

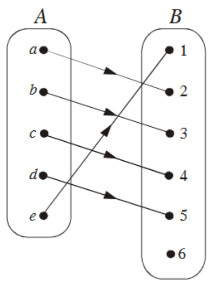
$$D_{i_A} = H_{i_A} = A$$

[7]

Injekcia. Zobrazenie $f : A \to B$ je injektívne vtedy a len vtedy, ak

$$\forall(x, y \in A; x \neq y) : (f(x) \neq f(y))$$

Obr. 1.12: Injektívne zobrazenie $f : A \to B$.



[7]

Surjekcia. Zobrazenie $f : A \to B$ je surjektívne vtedy a len vtedy, ak

$$\forall(y \in B)\exists(x \in A) : (f(x) = y)$$

[6]

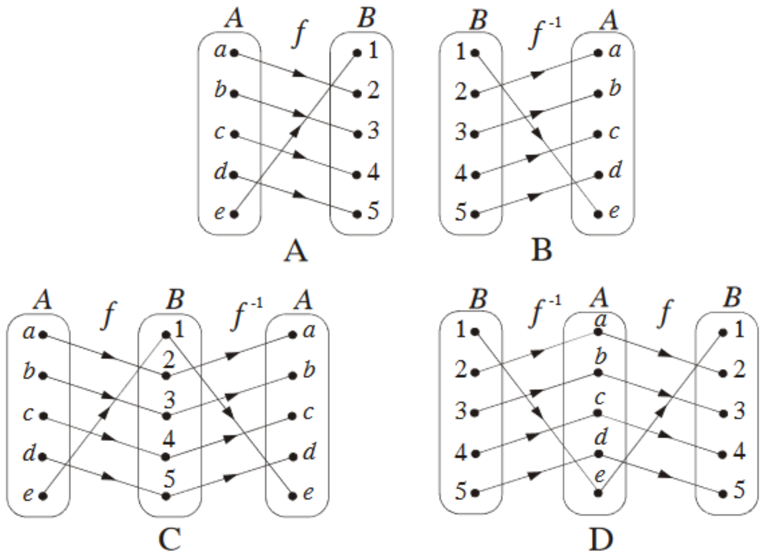
Bijekcia. Zobrazenie $f : A \to B$ je bijektívne vtedy a len vtedy, ak je súčasne injektívne aj surjektívne. [6]

Inverzné zobrazenie. Nech zobrazenie $f : A \to B$ je bijekcia, zobrazenie $f^{-1} : B \to A$ je inverzné zobrazenie k zobrazeniu f vtedy a len vtedy, ak súčasne platí

$$f(f^{-1}(x)) = i_B(x)$$

$$f^{-1}(f(x)) = i_A(x)$$

kde i_A (i_B) je jednotková funkcia definovaná nad množinou A (B). [7]

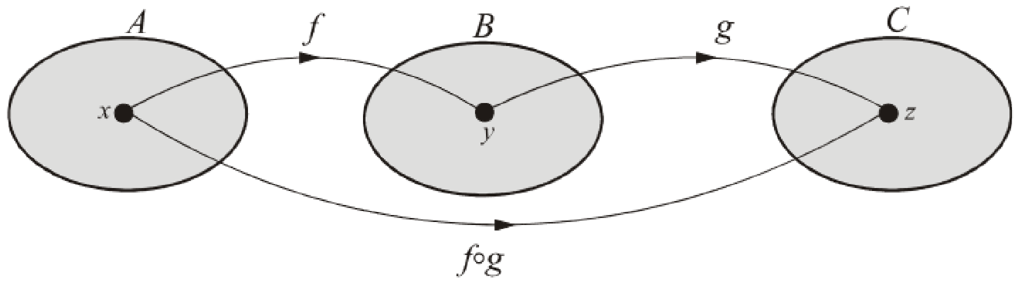


Obr. 1.13: Diagramy A a B znázorňujú zobrazenia f a f^{-1}. Diagram C znázorňuje kompozíciu zobrazení $f \circ f^{-1} = i_A : A \to A$, diagram D znázorňuje kompozíciu zobrazení $f \circ f^{-1} = i_B : B \to B$.

Kompozícia zobrazení. Nech $f : A \to B$ a $g : B \to C$ sú zobrazenia, potom $f \circ g : A \to C$ je kompozícia zobrazení f a g vtedy a len vtedy, ak

$$f \circ g = \{(x,z) \in A \times C; \exists (y \in B)((x,y) \in f) \wedge ((y,z) \in g)\}$$

[7]



Obr. 1.14: Znázornenie tvorby zloženého zobrazenia $f \circ g$ zo zobrazení f a g. Táto kompozícia zobrazení existuje len vtedy, keď prienik oboru funkčných hodnôt H_f zobrazenia f a definičného oboru D_g zobrazenia g je neprázdny, $H_f \cap D_g \neq \emptyset$.

2 Metodika práce a metódy skúmania

...

2.1 Charakteristika objektu skúmania

...

2.2 Súčasný stav riešeniej problematiky doma a v zahraničí

...

2.3 Cieľ práce

...

2.4 Pracovné postupy

...

3 Implementácia

Táto kapitola slúži na detailný popis samotnej implementácie riešenia pre softvérovej podpory vyučovania diskrétnej matematiky.

3.1 Abstraktné dátové typy

Vytvorenie abstraktných dátových typov pre binárne relácie, relácie na množine, čiastočné usporiadania a zobrazenia. Oddelenie a zjednodušenie programu pre prehľadnejšiu prácu s danými matematickýmy štruktúrami a pre uľahčenie testovania vlastností týchto štruktúr.

3.1.1 Binárne relácie

Zadefinovanie triedy BinaryRelation ako abstraktný dátový typ.

```
class BinaryRelation:
 def __init__(self,relation=None,domain=None,codomain=None,check=False):
  if check == True:
   if self.is_valid(relation, domain, codomain) == False:
    raise Exception
    (f"Binary relation {(relation, domain, codomain)} is not valid!")
  self._relation = FiniteSet(*relation)
  self._domain = FiniteSet(*domain)
  self._codomain = FiniteSet(*codomain)

 def is_valid(self, relation, domain, codomain):
  if len(relation) == 0:
   return True
  return FiniteSet(*relation) <= (FiniteSet(*domain) * FiniteSet(*codomain))
```

3.1.2 Relácie na množine

...

3.1.3 Čiastočné usporiadania

...

3.1.4 Zobrazenia

...

3.2 Reprezentácia matematických štruktúr

...

3.2.1 Množinová reprezentácia

...

3.2.2 Maticová reprezentácia

...

3.2.3 Grafová reprezentácia

...

3.3 Jazyk dopytov

...

3.4 Vysvetľovač

...

Záver

Literatúra

[1] The Graphviz Authors. What is graphviz? *graphviz.se*.

[2] Peter C.Fishburn. Operations on binary relations. *Discrete Mathematics*, 21:7–22, 1978.

[3] Niklas Eén and Niklas Sörensson. Introduction to minisat. *minisat.se*.

[4] Pablo Galindo. Python 3.10.0. *Python Insider*, 2021.

[5] Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser. *Data Structures and Algorithms in Python*. Wiley, 2013.

[6] Ralph P. Grimaldi. *Discrete and combinatorial mathematics: An applied introduction*. Boston: Pearson/Addison-Wesley, 2004.

[7] Vladimír Kvasnička and Jiří Pospíchal. *Algebra a diskrétna matematika*, volume 1. Slovenská technická univerzita, 2008.

[8] Olga Ohrimenko, Peter Stuckey, and Michael Codish. Principles and practice of constraint programming. *Lecture Notes in Computer Science*, 2007.

[9] R.L.Graham, D.E.Knuth, and T.S.Motzkin. Complements and transitive closures. *Discrete Mathematics*, 2:17–29, 1972.

[10] Cyrille Rossant. *Learning Ipython for Interactive Computing and Data Visualization*, volume 2. Packt Publishing, 2015.

[11] Al Sweigart. Automate the boring stuff with python. *No Starch Press*, 2015.

[12] Pavel Tišnovský. Jupyter notebook – nástroj pro programátory, výzkumníky i lektory. *Root.cz*, 2021.

[13] Pavel Tišnovský. Jupyter notebook – operace s rastrovými obrázky a uml diagramy, literate programming. *Root.cz*, 2021.