UNIVERZITA MATEJA BELA V BANSKEJ BYSTRICI
FAKULTA PRÍRODNÝCH VIED

Využitie podporného softvéru vo vyučovaní diskrétnej matematiky

DIPLOMOVÁ PRÁCA

57c45833-d235-445c-b4e9-0e98bff84c73

2015 Bc. Štefan Ušák

UNIVERZITA MATEJA BELA V BANSKEJ BYSTRICI
FAKULTA PRÍRODNÝCH VIED

Využitie podporného softvéru vo vyučovaní diskrétnej matematiky

Diplomová práca

57c45833-d235-445c-b4e9-0e98bff84c73

Študijný program:	Aplikovaná informatika
Študijný odbor:	9.2.9. aplikovaná informatika
Pracovisko:	KIN FPV - Katedra informatiky
Vedúci diplomovej práce:	PaedDr. Ivan Brodenec PhD.

Banská Bystrica, 2015 Bc. Štefan Ušák

Univerzita Mateja Bela v Banskej Bystrici
Fakulta prírodných vied

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Štefan Ušák
Študijný program:	aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	9.2.9. aplikovaná informatika
Typ záverečnej práce:	Magisterská záverečná práca
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Využitie podporného softvéru vo vyučovaní diskrétnej matematiky

Anotácia: Vytvorte rozšírenie svojej bakalárskej práce doplnené o ďalšie témy z vyučovania predmetu "Diskrétna matematika". Analyzujte jednotlivé algoritmy používané v teórii zväzov a kongruenciách na zväzoch. Overte aplikáciu na študentoch predmetu a získajte tak spätnú väzbu ako aplikácia slúži ako pomôcka v predmete.

Vedúci:	PaedDr. Ivan Brodenec, PhD.
Katedra:	KIN FPV - Katedra informatiky
Vedúci katedry:	PaedDr. Mgr. Vladimír Siládi, PhD.
Dátum zadania:	07.04.2014
Dátum schválenia:	19.03.2015

prof. Ing. Miroslav Svítek, Dr.
garant študijného programu

Čestné vyhlásenie

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím uvedených zdrojov.

V Banskej Bystrici, dňa 29.4.2015 .....

Poďakovanie

Ďakujem vedúcemu diplomovej práce, PaedDr. Ivanovi Brodencovi PhD., za pripomienky, rady a čas, ktorý mi poskytol.

Abstrakt

UŠÁK, Štefan: Využitie podporného softvéru vo vyučovaní diskrétnej matematiky. [Diplomová práca] / Štefan Ušák. – Univerzita Mateja Bela v Banskej Bystrici. Fakulta prírodných vied; Katedra informatiky. – Školiteľ: PaedDr. Ivan Brodenec PhD. Banská Bystrica FPV UMB, 2015, 57s.

Práca je pokračovaním bakalárskej práce. Rozširuje ju o ďalšie dve témy z diskrétnej matematiky (ekvivalencie a kongruencie). Vytvorený program umožňuje z úvodnej LP tabuľky príkladu v pár krokoch vygenerovať LP tabuľku uzáverov. Nájdené uzávery následne vykresliť do Hasseovej diagramu. Hľadanie uzáverov je možné manuálne alebo automaticky stlačením tlačidla. V práci sa ďalej nachádza zhodnotenie spätnej väzby od študentov, ktorí program používali v praxi.

Kľúčové slová

Ekvivalencie, kongruencie, LP tabuľka, uzáver, Hasseovej diagram, algoritmus

Abstrakt

UŠÁK, Štefan: The use of supporting software in Discrete Mathemathics teaching. [Master thesis] / Štefan Ušák. – Matej Bel University in Banská Bystrica. Faculty of natural sciences, Department of computer science. – Supervisor: PaedDr. Ivan Brodenec PhD.. Banská Bystrica FPV UMB, 2015, 57p.

This master thesis extends the work done in the bachelor thesis. Two themes from discrete mathematics are added. A program which allows to generate an LP table of closure is created. Closures are then displayed in Hasse diagram. This can be done automatically or manually by pushing a button. Moreover, the work obtains valuation of students, who used the program practically.

Key words:

Equivalencies, congruencies, LP table, closure, Hasse diagram

Predhovor

Každý študent aplikovanej informatiky, na Univerzite Mateja Bela v Banskej Bystrici, sa v prvom ročníku bakalárskeho štúdia stretne s predmetom diskrétna matematika. Nie každému študentovi sa podarí týmto predmetom prejsť s ľahkosťou. Z toho dôvodu vznikol podnet na vytvorenie učebnej pomôcky, ktorá študentom uľahčí tento predmet.

Vytvorená učebná pomôcka by mala pomôcť k lepšiemu pochopeniu časti preberaného učiva diskrétnej matematiky. Diplomová práca je rozšírením programu bakalárskej práce, ktorý sa zameriaval na tému generovanie zväzu zdola nahor. Poskytuje aj spätnú väzbu používania programu študentmi.

Účelom použitia učebnej pomôcky je uľahčiť študentom pochopenie daných tém, ale aj dosahovať lepšie výsledky v predmete diskrétna matematika. Tento program je vhodný nielen pre študentov, ale aj pre profesorov.

Obsah

Zoznam obrázkov

Zoznam grafov

Špecifikácia pojmov

Pred písaním práce som sa musel rozhodnúť, či budem používať pojem Hasseho, alebo Hasseovej diagram. V príkladoch som sa stretol s oboma pojmami. Priklonil som sa k pojmu Hasseovej diagram a to z dôvodu, že sa nachádza aj v literatúre, z ktorej som vychádzal.

Úvod

S predmetom diskrétna matematika sa stretnú študenti v prvom ročníku bakalárskeho stupňa štúdia. V rámci bakalárskej práce *Demonštračný program pre diskrétnu matematiku* som spracoval tému generovanie zväzu zdola nahor a vytvoril program. Vytvorený program umožnil študentom zo vstupnej LP tabuľky zadania vygenerovať výstupnú LP tabuľku uzáverov. Študent môže LP tabuľku generovať automaticky stlačením tlačidla alebo postupne hľadaním uzáverov jednotlivým podmnožinám z konečnej množiny. Pri postupnom hľadaní uzáverov vie program skontrolovať správnosti riešenia a upozorniť na vzniknuté chyby. Ak chce študent v riešení príkladu pokračovať, musí vzniknuté chyby odstrániť. Ak sa mu to nepodarí, má možnosť si dogenerovať tabuľku automaticky. Z tabuľky uzáverov má študent možnosť nakresliť Hasseovej diagram. Ako pri hľadaní uzáverov, aj teraz má možnosť kresliť automaticky stlačením tlačidla alebo manuálne. Pri manuálnom kreslení program umožňuje výsledný diagram skontrolovať. Ak sa v diagrame nachádzajú chyby, upozorní na ne. Ak je diagram správny, oznámi študentovi túto skutočnosť. Vytvorený program nie je vhodný len pre študentov, ale aj pre vyučujúcich, ktorí učia diskrétnu matematiku. Keďže umožňuje automatické riešenie príkladov, vedia rýchlo nagenerovať rôzne ťažké príklady na skúšku za pár sekúnd.

Prvým cieľom diplomovej práce je rozšíriť spomínaný program o dve algebry: ekvivalencie a kongruencie. Študenti budú mať možnosť vybrať si jednu z troch tém. Pre zvolenú tému by mal program umožniť rovnakú funkcionalitu, akú umožňuje pri téme generovanie zväzu zdola nahor. Pri ekvivalenciách sa tiež vychádza z úvodnej LP tabuľky a výsledkom je Hasseovej diagram. Kongruencie sú zadávané binárnymi alebo unárnymi operáciami, z ktorých sa následne vygeneruje LP tabuľka príkladu. Výsledkom je tiež Hasseovej diagram. Každej kongruencie, ktorá sa nachádza v Hasseovej diagrame, treba nájsť blokové zápisy faktorových algebier. Program bude mať pri kongruenciách rozšírenú funkcionalitu, aby študent nebol nijak obmedzený. Podmnožiny pri témach ekvivalencie a kongruencie môžu obsahovať lomeno. Pri téme generovania zväzu zdola nahor podmnožiny neobsahovali lomeno. Program bude musieť byť prispôsobený aj na túto skutočnosť.

Druhým cieľom práce je zistiť, či vytvorený program bude mať medzi študentmi, pre ktorých je určený, pozitívne, alebo negatívne hodnotenie. Na splnenie druhého cieľa bude treba najskôr splniť prvý cieľ. Následne bude potrebné vytvoriť krátky dotazník na spätnú väzbu od študentov. Vytvorený program aj s dotazníkom sa poskytne študentom. Okrem toho

bude študentom prístupná aj používateľská príručka. V príručke je prebraná funkcionalita programu na konkrétnych príkladoch. Na konci príručky sa nachádza zopár vzorových príkladov ku každej téme.

Pri rozšírení programu zužitkujem informácie nadobudnuté od vyučujúceho Mgr. Michala Vagača PhD. Pri písaní práce budem vychádzať zo svojej bakalárskej prace, študentskej vedeckej činnosti a zo skrípt Doc. RNDr. Bohuslava Siváka, CSc.

1 Algebry

V prvej kapitole budem vychádzať zo skrípt Doc. RNDr. Bohuslava Siváka, CSc. Diskrétna matematika, konkrétne strany 43 až 65.

Univerzálna algebra je taká dvojica (A,F), kde A je spravidla konečná množina a F je podsystém operácií ľubovoľnej árnosti na konečnej množine A. Predstavme si množinu A a celé nezáporne číslo n. Symbolom A^n sa potom označuje množina všetkých usporiadaných n-tíc prvkov množiny A. N-árna operácia na množine A je ľubovoľná funkcia $A^n \rightarrow A$. Číslo n nám určuje árnosť operácie. Pre malé hodnoty n sa používajú špeciálne názvy operácií:

n = 0 ..... takáto operácia sa nazýva nulárna,

n = 1 ..... takáto operácia sa nazýva unárna,

n = 2 ..... takáto operácia sa nazýva binárna,

n = 3 ..... takáto operácia sa nazýva ternárna.

V programe budeme pracovať s unárnymi a binárnymi operáciami.

Zväzy pri téme generovanie zväzu zdola nahor môžeme chápať ako algebry s dvoma binárnymi operáciami. Takéto algebry sú typu (2,2). Algebra, ktorá má len jednu a to binárnu operáciu, sa označuje grupoid. Grupoidom sa označuje ľubovoľná algebra typu (2).

Ako príklad uzavretosti množiny na operáciu si predstavme n-ármu operáciu op na množine A a podmnožinu $X \subseteq A$. Množina X je uzavretá na operáciu, ak platí:

$$x_1 \in X,\ x_2 \in X, \ldots\ldots, x_n \in X \rightarrow op(x_1, x_2, \ldots\ldots, x_n) \in X.$$

Ako druhý príklad, pre lepšie pochopenie uzavretosti množiny, si predstavme množinu Z, ktorá obsahuje všetky celé čísla vrátane záporných a nuly, X je množina všetkých párnych čísel a Y všetkých nepárnych čísel. Potom môžeme povedať, že X je uzavretá na operáciu súčtu, rozdielu aj súčinu, no množina Y je uzavretá len na operáciu súčinu.

Podmnožina algebry sa nazýva podalgebra za predpokladu, že je uzavretá na všetky operácie algebry. "*Systém všetkých podalgebier danej algebry vždy tvorí množinový zväz.* "[2] Tento zväz môžeme hľadať podobne ako pri téme generovanie zväzu zdola nahor, ak dokážeme pre danú algebru zostaviť adekvátnu LP-tabuľku. Ak algebra nemá žiadne nulárne operácie, množinový zväz podalgebier obsahuje aj prázdnu množinu.

Pri zostavovaní LP tabuľky pre hľadanie podalgebier treba dbať na to, aby uzavretými množinami boli práve všetky podalgebry danej algebry. Pre každú n-árnu operáciu op treba

nájsť také n-tice ($x_1, x_2, \ldots\ldots, x_n$), pre ktoré platí $op(x_1, x_2, \ldots\ldots, x_n) \notin \{X_1, X_2, \ldots\ldots, X_n\}$. Následne do LP tabuľky naľavo zapíšeme množinu $\{X_1, X_2, \ldots\ldots, X_n\}$ a na pravú stranu prvok $op(x_1, x_2, \ldots\ldots, x_n)$. V prípade, že máme riadky s rovnakou L-množinou, je možné zlúčiť ich do jedného tak, že ich P-množiny zjednotíme.

Postup vytvárania LP tabuľky sa dá ľahšie pochopiť na konkrétnych príkladoch. "*Každá algebra je sama algebrou toho istého typu a podalgebry môžeme popísať podobnými tabuľkami ako pôvodnú algebru.* "[2]

1.1 Podalgebry unárnych a binárnych algebier

V informatike sa prakticky nestretneme s algebrou, ktorá by mala viacej ako dve operácie (binárna algebra). V ďalšej časti si v krátkosti priblížime unárne a binárne algebry, s ktorými vie program pracovať.

1.1.1 Podalgebry unárnych algebier

Predstavme si unárnu operáciu (A, Φ), kde A = {a, b, c, d} a operácia Φ je znázornená na obrázku č. 1.

	Φ
a	b
b	b
c	c
d	a

Obrázok č. 1: Príklad č. 1 operácia Φ

Z tabuľky na obrázku č. 1 vytvoríme novú tabuľku. V novej tabuľke ponecháme len operácie, ktoré majú ľavú stranu rozdielnu od pravej. Ako vidno na obrázku č 2., budú nás zaujímať iba „nové" výsledky operácie.

	Φ
a	b
b	—
c	—
d	a

Obrázok č. 2: Príklad č. 1 výsledky operácia Φ

Z obrázku č. 2 vytvoríme úvodnú LP tabuľku príkladu. LP tabuľka sa nachádza na obrázku č.3.

L	P
a	b
d	a

Obrázok č. 3: Úvodná LP tabuľka príkladu č. 1

Ak uprednostníme generovanie zväzu zhora nadol, postup bude nasledovný:

[a] = ab,

[b] = b,

[c] = c,

[d] = abd.

[a] + [c] = [a, c] = abc,

[b] + [c] = [b, c] = bc,

[c] + [d] = [c, d] = A.

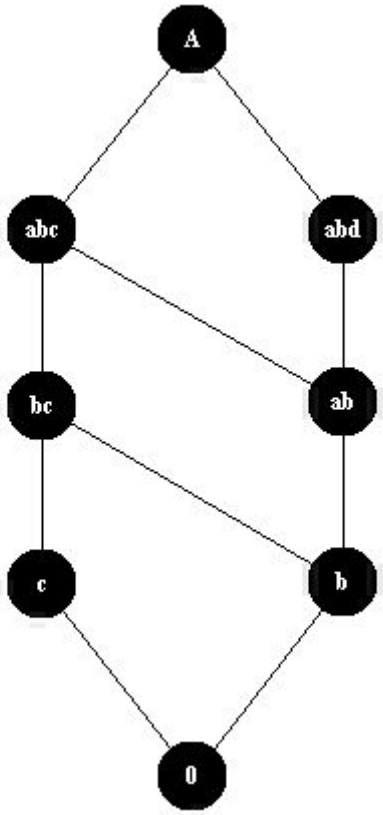
Pri kontrole neporovnateľných prvkov zistíme, že nám žiadne nové súčty nevznikli.

[a] + [b, c] = [a, b, c] = abc = [a, c],

[d] + [b, c] = A,

[d] + [b, c] = A.

Výsledný Hasseovej diagramom sa nachádza na obrázku č 4.



Obrázok č. 4: Výsledný Hasseovej diagram príkladu č. 1

Na obrázku č 5. sa nachádzajú zápisy vzniknutých podalgebier.

	Φ
a	b
b	b

	Φ
b	c
c	c

	Φ
a	b
b	b
c	c

	Φ
a	b
b	b
d	a

Obrázok č. 5: Zápisy operácií príkladu č. 1

1.1.2 Podalgebry binárnych algebier

Predstavme si grupoid (A,.), kde A = {a, b, c} a operáciu Φ, tentokrát danú Cayleyho tabuľkou na obrázku č. 6.

.	a	b	c
a	a	c	a
b	b	b	c
c	a	c	a

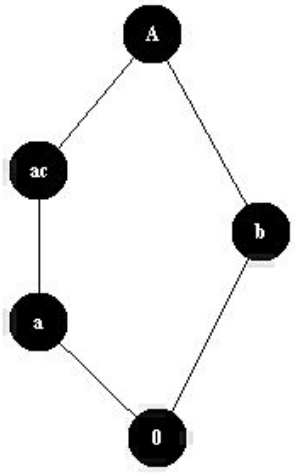
Obrázok č. 6: Cayleyho tabuľka príkladu č. 2

Z Cayleyho tabuľky vytvoríme novú tabuľku. Nová tabuľka sa nachádza na obrázku č. 7 a obsahuje iba nové výsledky operácií.

.	a	b	c
a	—	c	—
b	—	—	—
c	—	—	a

Obrázok č. 7: Výsledky operácie z Cayleyho tabuľky príkladu č. 2

Z danej tabuľky vytvoríme Hasseovej diagram, ktorý bude obsahovať iba jednu dvojprvkovú podalgebru. Hasseovej diagram je znázornený na obrázku č. 8. K podalgebre treba nájsť jej zápisy. Zápis sa taktiež nachádza na obrázku č. 8.



.	a	c
a	a	a
c	a	a

Obrázok č. 8: Hasseovej diagram a zápis podalgebry príkladu č. 2

1.2 Ekvivalencie

Ekvivalencia na množine A je taká binárna relácia na množine A, ktorá je reflexívna, symetrická a taktiež tranzitívna. Systém všetkých ekvivalencií na konečnej množine A tvorí množinový zväz. Základnou množinou zväzu je množina AxA. Ako sme už zvyknutí, zväzovým súčinom je množinový prienik. Predstavme si konečnú množinu A, zloženú z dvoch prvkov A = {1, 2}. Na tejto množine nájdeme len dve ekvivalencie:

- diagonála (diag) ..... {(1,1), (2,2)},
- najväčšia (AxA) ……..{(1,1), (1,2), (2,1), (2,2)}.

Takýto zápis vymenovaním prvkov je nepohodlný pri väčšom počte prvkov v konečnej množine. Jednoduchšie je ekvivalencie rozkladať pomocou takzvaných blokov rozkladu. Bloky rozkladu sú také pokrytia množiny A, v ktorých žiadne dve množiny nemajú žiaden spoločný prvok. Pri zápise rozkladu môžeme použiť buď úplný blokový zápis alebo skrátený blokový zápis. Pri skrátenom blokovom zápise sa ignorujú všetky jednoprvkové bloky. Úplný blokový zápis môže vyzerať nasledovne: [13][2], [1][23], [13][25][4], atď. Skrátené zápisy týchto ekvivalencií sú nasledovné: 13, 23 a 13/25. Tieto skrátene zápisy sa používajú hlavne pri kreslení Hasseovej diagramu, kde by úplný blokový zápis zaberal príliš veľa miesta.

Prienikom dvoch ekvivalencií je opäť ekvivalencia, ktorej blokmi sú neprázdne prieniky blokov daných ekvivalencií. Výsledok vychádza správne pri oboch typoch blokového zápisu. Príklad prieniku ekvivalencií v skrátenom blokovom zápise vyzerá nasledovne:

$134 \cap 13 = 13,$

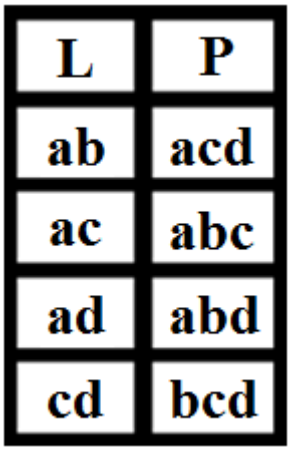
$25 \cap 24 = 2,$

$25 \cap 5 = 5.$

Symbolom Eq(A) sa označuje zväz všetkých ekvivalencií na množine A. Ako pri prieniku, aj pri výpočte súčtu vo zväze môžeme počítať pomocou blokového zápisu. Postup je veľmi jednoduchý. Predstavme si, že máme blokové zápisy dvoch ekvivalencií. Napíšeme ich vedľa seba a opakujeme nasledujúci krok. Ak majú dva bloky spoločný aspoň jeden prvok, tak tieto bloky nahradíme ich zjednotením. Ak už nenájdeme spoločné prvky, máme blokový zápis súčtu.

Pri ekvivalenciách vyzerá LP tabuľka príkladu inak ako pri téme generovania zväzu zdola nahor. V ľavom stĺpci sú vždy dva prvky z konečnej množiny A. V pravom stĺpci je vždy skrátený blokový zápis ekvivalencie. Takáto tabuľka sa nazýva tabuľkou kompatibility. Na obrázku č. 9 sa nachádza LP tabuľka vzorového príkladu. Konečná množina A sa skladá

zo 4 prvkov, A = {a, b, c, d}. Ďalej si ukážeme, ako z LP tabuľky zadania nájsť uzávery ekvivalenciám.



L	P
ab	acd
ac	abc
ad	abd
cd	bcd

Obrázok č. 9: Zadanie vzorového príkladu na ekvivalencie

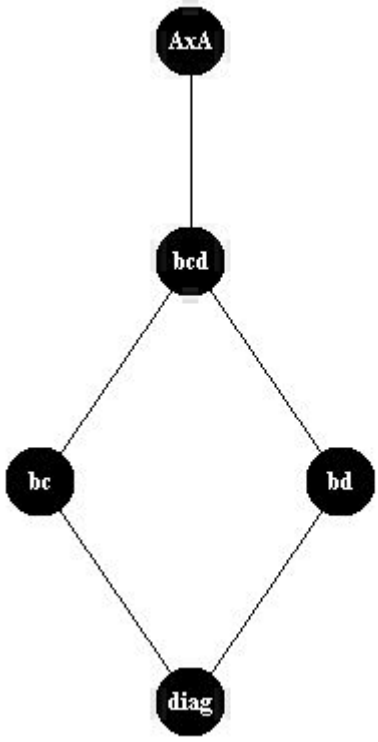
Algoritmus hľadania uzáverov ekvivalenciám je podobný ako pri téme generovania zväzu zdola nahor. V tomto prípade začíname dvojprvkovými ekvivalenciami. Do novej LP tabuľky si napíšeme všetky dvojprvkové ekvivalencie z konečnej množiny. Ak sa podmnožina v LP tabuľke zadania nenachádza, jej uzáverom je ona sama, tak ako pri téme generovanie zväzu zdola nahor. V opačnom prípade musíme nájsť uzáver ekvivalencie podľa zadania. Ako príklad si zoberme ekvivalenciu ab[1]. Na pravej strane tabuľky nájdeme trojprvkovú ekvivalenciu acd. Z tejto ekvivalencie nám vzniknú ďalšie nové, ktoré musíme taktiež brať do úvahy. V ekvivalencii acd sa objavili nové ekvivalencie ac, ad a cd. Uzáver k ekvivalencii ab by vyzeral nasledovne:

$$acd + abc + abd + bcd = AxA.$$

V tomto prípade nevznikli žiadne neporovnateľné prvky. Výsledná LP tabuľka aj s Hasseovej diagramom sa nachádza na obrázku č. 10.

[1] V literatúre na môže ekvivalencie ab nachádzať aj v tvare a,b.

L	P
ab	AxA
ac	AxA
ad	AxA
bc	bc
bd	bd
cd	bcd



Obrázok č. 10: Výsledná LP tabuľka s Hasseovej diagramom vzorového príkladu

1.3 Kongruencie

Všetky ekvivalencie sú kompatibilné so všetkými konštantnými operáciami. Pri unárnych algebrách budú dôležité iba prípady, keď budeme mať len jednu alebo viac unárnych operácií. Tieto operácie budú v tabuľke zapísané ako stĺpce. Pri binárnych algebrách budú operácie zapísané v Cayleyho tabuľke. Na príkladoch ukážem hľadanie tabuľky kompatibility a uzáverov pre unárne aj binárne operácie.

Majme unárnu algebru (A, Φ_1, Φ_2), kde operácie sú znázornené na obrázku č. 11.

	Φ_1	Φ_2
a	b	d
b	b	b
c	c	c
d	a	d

Obrázok č. 11: Operácie unárnej algebry príkladu č. 4

Z tabuľky na obrázku č. 11 zostavíme tabuľku kompatibility. Tabuľku kompatibility zostavujeme tak, že pre každú dvojicu vezmeme do úvahy aj dvojice hodnôt stĺpcov. Pre dvojicu ab by to vyzeralo nasledovne:

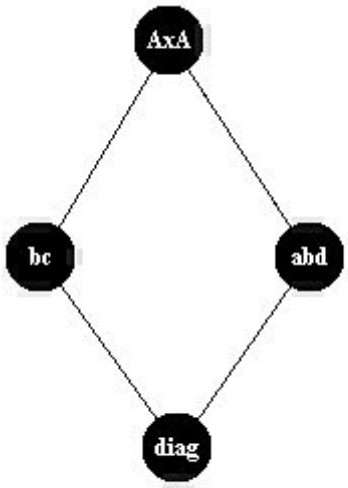
ab + bb + db = adb.

Obdobným spôsobom prejdeme všetky dvojice prvkov a výsledné podmnožiny zapíšeme do novej LP tabuľky. Podľa tejto LP tabuľky nájdeme uzávery podmnožinám. Tabuľka kompatibility aj s uzávermi k jednotlivým podmnožinám sa nachádzajú na obrázku č. 12.

L	P	uzáver
ab	abd	abd
ac	AxA	AxA
ad	abd	abd
bc	bc	bc
bd	abd	abd
cd	acd	AxA

Obrázok č. 12: LP tabuľka a uzávery k unárnej algebre príkladu č. 4

Vzniknuté uzávery nakreslíme do Hasseovej diagramu. Hasseovej diagram sa nachádza na obrázku č. 13.



Obrázok č. 13: Hasseovej diagram príkladu č. 4

Pri každej kongruencii môžeme vytvoriť takzvanú faktorovú algebru. Faktorovú algebru vytvoríme nasledovne:

- napíšeme úplný blokový zápis kongruencie
- z každého bloku vyberieme jeden prvok, ktorý budeme nazývať reprezentant
- vytvoríme novú tabuľku pre operácie, do záhlavia napíšeme iba reprezentantov, vo výsledku urobíme nasledujúcu zmenu a ak výsledok nie je reprezentant, tak sa pozrieme, do akého bloku patrí, a výsledok nahradíme reprezentantom toho bloku.

V našom prípade vznikli vo výsledku štyri kongruencie. Ak by sme faktorizovali podľa diagonály, dostali by sme pôvodnú tabuľku. Pri faktorizovaní podľa AxA by vznikla jednoprvková algebra. Z toho dôvodu nás budú zaujímať len posledné dve kongruencie. Blokové zápisy faktorových algebier s operáciami pre kongruencie abd a bc sa nachádzajú na obrázku č. 13.

abd = [a, b, d][c]

	Φ_1	Φ_2
a	a	a
c	c	c

bc = [a][b, c][d]

	Φ_1	Φ_2
a	a	d
b	b	b
d	a	d

Obrázok č. 13: Blokové zápisy faktorových algebier príkladu č. 4

Predstavme si teraz binárnu algebru danú Cayleyho tabuľkou. Cayleyho tabuľka sa nachádza na obrázku č. 14.

.	a	b	c	d
a	a	a	b	a
b	b	a	b	a
c	c	b	a	b
d	d	b	a	b

Obrázok č. 14: Cayleyho tabuľka pre binárnu algebru príkladu č. 5

Z tejto tabuľky vytvoríme tabuľku kompatibility podobným spôsobom ako v predchádzajúcom príklade. Tabuľka sa nachádza na obrázku č. 15.

L	P
ab	AxA
ac	AxA
ad	AxA
bc	abc
bd	abd
cd	ab/cd

Obrázok č. 15: LP tabuľka kompatibility príkladu č. 5

O triedach ekvivalencií, ktoré obsahujú viac tried ekvivalencií (množina ab/cd) sa v programe budem vyjadrovať ako o podmnožinách obsahujúcich lomeno.

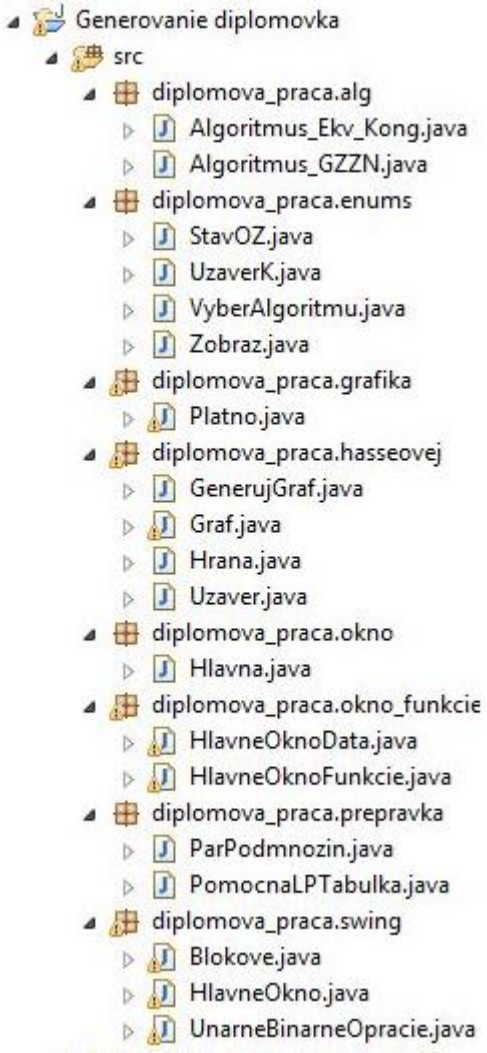
Podľa novej tabuľky nájdeme uzávery jednotlivým podmnožinám. Výsledná LP tabuľka aj s Hasseovej diagramom sa nachádzajú na obrázku č. 16. Vznikli nám len dve kongruencie, pre ktoré sa neoplatí zostavovať blokové zápisy.

L	P
ab	AxA
ac	AxA
ad	AxA
bc	AxA
bd	AxA
cd	AxA

Obrázok č. 16: LP tabuľka uzáverov a Hasseovej diagram príkladu č. 5

2 Rozšírenie programu

Bakalársky program prešiel mnohými zmenami. Okrem potrebného rozšírenia o triedy `Algoritmus_Ekv_Kong`, `VyberAlgoritmu`, `PomocnáLPTabulka`, `Blokove` a `UnarneBinarneOperacie`, bol rozdelený namiesto dvoch balíčkov do siedmych. Tento počet balíčkov som zvolil z dôvodu lepšej prehľadnosti štruktúry programu. Hierarchia tried programu sa nachádza na obrázku č. 17. Do tried `GenerujGraf` a `HlavneOknoFunkcie` bolo treba pridať viac funkcionality ako do iných tried. Trieda `GenerujGraf` pracujú presne tak isto ako som popisoval v bakalárskej práci, len s tým rozdielom, že ak ide o tému ekvivalencie alebo kongruencie, nerátajú s jednoprvkovými podmnožinami. Triedu bolo treba prispôsobiť tak, aby sa isté metódy volali len pri prvej téme a iné pri druhých dvoch. Podstatnou zmenou v triede `HlavneOknoFunkcie` bolo to, že v LP tabuľkách sa objavovali dvoj a viac prvkové podmnožiny. Podľa toho sa museli nadstaviť vstupné polia. Zmeny v iných triedach boli len drobné, väčšinou išlo o opravu kódu. Okrem funkcionality bolo potrebné upraviť aj okno programu. Pribudol výber z tém a dve tlačidlá: *Unárne/Binárne operácie* a *Blokové zápisy*. Algoritmy sa vyberajú s enumu `VyberAlgoritmu`. Nové tlačidlá sa viažu na tému kongruencie a ich funkcionalita je vysvetlená v prílohe B – používateľská príručka.



Obrázok č. 17: Hierarchia tried programu

Ako vidno na obrázku č. 17, v jednotlivých balíčkoch sa nachádzajú triedy, ktoré spolu súvisia. Názov balíčku je výstižný a naznačuje, aké triedy sa v ňom nachádzajú. Napríklad balíček grafika obsahuje triedu `Plátno`, ktorá slúži na vykreslenie Haseovej diagramu. Nová trieda `PomocnáLPTabulka` symbolizuje jeden riadok v pomocnej LP tabuľke. V triede sa nachádza parametrický konštruktor, v ktorom sa nastavia indexy riadku v tabuľke. Ďalej sa v práci budem venovať novým triedam v programe, ktoré neboli opísané v bakalárskej práci.

2.1 Trieda Algoritmus_Ekv_Kong

Trieda `Algoritmus_Ekv_Kong` je modifikovaná trieda `Algoritmus`, ktorú som podrobne opísal v bakalárskej práci. [1] V diplomovej práci sa táto trieda nachádza pod názvom `Algoritmus_GZZN`. Metódy z triedy `Algoritmus_Ekv_Kong` sú volané, ak používateľ pracuje s ekvivalenciami alebo kongruenciami.

Obe triedy majú za úlohu z LP tabuľky zadania vygenerovať LP tabuľku uzáverov. V ArrayListe `LPTabulkaVygenerovana` sa na začiatku generovania nachádzajú len podmnožiny z konečnej množiny A, ku ktorým sa budú hľadať uzávery. V triede `Algoritmus_GZZN` sú to zväčša jedno a dvojprvkové podmnožiny. Téma generovanie zväzu zdola nahor nepredpokladá vo vstupných podmnožinách lomeno. Z toho dôvodu bolo hľadanie uzáverov jednoduchšie ako pri ďalších dvoch témach. V triede `Algoritmus_Ekv_Kong` sa na začiatku generovania v ArrayListe `LPTabulkaVygenerovana` nachádzajú dvojprvkové, prípadne viac prvkové podmnožiny zo zadania príkladu. Dvojprvkové podmnožiny sú generované automaticky programom. Vo viacprvkových podmnožinách zo zadania sa môže nachádzať lomeno. Taktiež sa môže stať, že dvojprvková podmnožina sa rozširuje o podmnožinu obsahujúcu lomeno. Na túto možnosť sa musel algoritmus generovania uzáverov pre kongruencie a ekvivalencie rozšíriť. Všetky podstatné zmeny v metódach si postupne priblížime. Pre ľahšie pochopenie metódy `algoritmus_ekv_kong` si ako prvú priblížime metódu `kontrolujDvojprvky`.

2.1.1 Metóda kontrolujDvojprvky

Metóda s rovnakým názvom sa nachádza aj v triede `Algoritmus_GZZN`. Rozdiel medzi nimi je ten, že v triede `Algoritmus_Ekv_Kong` metóda pracuje s dvomi vstupnými parametrami.

```
pom = prvok;

pom_dva = druhy_prvok;
```

Obrázok č. 18: Uloženie vstupných podmnožín v metóde kontrolujDvojprvky

Na začiatku si algoritmus uloží vstupné podmnožiny do pomocných reťazcov (obrázok č. 18). V ďalšej časti pracuje s prvou vstupnou podmnožinou (`prvok`). Podmnožinu si rozdelí na menšie dvojprvkové a postupne ich hľadá na ľavej strane v LP tabuľke zadania (ArrayList `LPTabulkaZadania`). Ak nájde zhodu, pozrie sa na pravú stranu. Ak pravá strana neobsahuje lomeno a nachádza sa v nej prvok, ktorý vstupná podmnožina ešte neobsahuje, tak ho do nej pridá. Túto funkcionalitu nájdeme aj v metóde `kontrolujDvojprvky` s jedným parametrom v triede `Algoritmus_GZZN`. Ak však pravá strana obsahuje lomeno, postup je zložitejší.

Ako prvé si musí rozdeliť podmnožinu zo zadania podľa lomena. Na obrázku č. 19 sa nachádza časť kódu z metódy, ktorá podmnožinu rozdelí do dvoch objektov triedy `StringBuffer`. Vzniknuté podmnožiny následne program prekontroluje znak po znaku so vstupnou podmnožinou.

```
final StringBuffer stringPoCiarku = new StringBuffer();
final StringBuffer stringOdCiarky = new StringBuffer();

boolean pomocna_poCiarku = true;

for(int m = 0; m < LPTabulkaZadania.get(j).getPravaStrana().length(); m++) {
    if(LPTabulkaZadania.get(j).getPravaStrana().charAt(m) != '/') {
        if(pomocna_poCiarku) {
            stringPoCiarku.append(LPTabulkaZadania.get(j).getPravaStrana().charAt(m));
        } else {
            stringOdCiarky.append(LPTabulkaZadania.get(j).getPravaStrana().charAt(m));
        }
    } else {
        pomocna_poCiarku = false;
    }
}
```

Obrázok č. 19: Ukážka zdrojového kódu s metódy kontrolujDvojprvky, rozdelenie podmnožiny podľa lomena

Na obrázku č. 20 vidíme kontrolu so vstupnou podmnožinou. Algoritmus prekontroluje, či sa vstupná podmnožina (`prvok`) zhoduje aspoň v jednom znaku s rozdelenými podmnožinami zo zadania. Ak áno, pomocnú podmnožinu `pom` rozšíri o podmnožinu zo zadania.

```
boolean nasolPoCarku = false;
boolean nasolOdCarky = false;

for(int a = 0; a < prvok.length(); a++) {
    for(int b = 0; b < stringPoCiarku.length(); b++) {
        if(prvok.charAt(a) == stringPoCiarku.charAt(b)) {
            nasolPoCarku = true;
        }
    }
    for(int b = 0; b < stringOdCiarky.length(); b++) {
        if(prvok.charAt(a) == stringOdCiarky.charAt(b)) {
            nasolOdCarky = true;
        }
    }
}

if(nasolPoCarku) {
    pom = hlavneOknoData.spojPrvky(pom, stringPoCiarku.toString());
}
if(nasolOdCarky) {
    pom = hlavneOknoData.spojPrvky(pom, stringOdCiarky.toString());
}
```

Obrázok č. 20: Ukážka zdrojového kódu s metódy kontrolujDvojprvky, rozšírenie reťazca pom

Rovnakým spôsobom treba prekontrolovať aj druhú vstupnú podmnožinu (obrázok č. 21).

```
nasolPoCarku = false;
nasolOdCarky = false;

for(int a = 0; a < druhy_prvok.length(); a++) {
    for(int b = 0; b < stringPoCiarku.length(); b++) {
        if(druhy_prvok.charAt(a) == stringPoCiarku.charAt(b)) {
            nasolPoCarku = true;
        }
    }
    for(int b = 0; b < stringOdCiarky.length(); b++) {
        if(druhy_prvok.charAt(a) == stringOdCiarky.charAt(b)) {
            nasolOdCarky = true;
        }
    }
}

if(nasolPoCarku) {
    pom_dva = hlavneOknoData.spojPrvky(pom_dva, stringPoCiarku.toString());
}
if(nasolOdCarky) {
    pom_dva = hlavneOknoData.spojPrvky(pom_dva, stringOdCiarky.toString());
}
```

Obrázok č. 21: Ukážka zdrojového kódu s metódy kontrolujDvojprvky, rozšírenie reťazca pom_dva

Takýmto spôsobom porozširuje vstupné podmnožiny podľa LP tabuľky zadania. Na záver metódy zistí, či sa prvá vstupná podmnožina (`prvok`) zmenila. Ak sa zmenila, treba opäť prekontrolovať všetky dvojprvkové podmnožiny. Preto je znovu volaná metóda `kontrolujDvojprvky`. Za predpokladu, že podmnožina sa rozšírila na veľkosť konečnej množiny A, je ďalšie hľadanie prvkov zbytočné. Uzáverom podmnožiny je celá množina A. Ak sa nezmenila, vráti nájdený uzáver (obrázok č. 22).

```
if(!pom.equals(prvok)) {
    prvok = pom;
    if(prvok.length() == hlavneOknoData.getVelkostZvazu()) {
        return prvok;
    }
    kontrolujDvojprvky(prvok, druhy_prvok);
}
prvok = pom;

return prvok;
```

Obrázok č. 22: Ukážka zdrojového kódu z metódy kontrolujDvojprvky, kontrola na konci metódy

2.1.2 Metóda algoritmus_ekv_kong

Metóda `algoritmus_ekv_kong` volá rôzne metódy, ktoré postupne vytvoria výslednú LP tabuľku uzáverov. V triede `Algoritmus_GZZN` túto činnosť vykonávala metóda `algoritmus_gzzn`.

Ako prvé metóda skontroluje, či sa podmnožina nachádza v LP tabuľke zadania. Bolo nutné rozlíšiť podmnožiny, ktoré obsahujú, a ktoré neobsahujú lomeno. V prípade, že podmnožina je bez lomena a v LP tabuľke zadania ju nájdeme s pravou stranou tiež bez

lomena, zlepíme tieto dve podmnožiny a vzniknutú podmnožinu bez duplicitných prvkov pošleme do metódy `kontrolujDvojprvky`. Ak podmnožina je bez lomena a nenachádza sa v LP tabuľke zadania, do metódy `kontrolujDvojprvky` sa pošle podmnožina samotná. Nelepí sa so žiadnou inou podmnožinou.

V týchto dvoch prípadoch pracuje metóda rovnako ako pri téme generovanie zväzu zdola nahor. Ako som už vyššie spomínal, v téme generovanie zväzu zdola nahor sa pracuje len s podmnožinami, ktoré neobsahujú lomeno. Pri ekvivalenciách a kongruenciách však musíme rátať aj s prípadmi, že metódy obsahujú lomeno. Z toho dôvodu bolo potrebné metódu `kontrolujDvojprvky` upraviť. Bola pridaná druhá vstupná podmnožina do metódy. V týchto dvoch prípadoch druhá podmnožina ostala prázdna. Metóda `kontrolujDvojprvky` je popísaná v predchádzajúcej podkapitole (2.1.1 Metóda kontrolujDvojprvky).

Ak algoritmus hľadania uzáverov pri prechádzaní ArrayListu `LPTabulkaVygenerovana` narazí na podmnožinu, ktorá sa v LP tabuľke zadania (ArrayList `LPTabulkaZadania`) nachádza, ale má na pravej strane lomeno, postupuje iným spôsobom ako v predchádzajúcich prípadoch. Ako prvé pravú stranu, s ktorou by podmnožinu lepil, keby nemala lomeno, rozdelí podľa lomena na dve. Presne tak ako na obrázku č 19. V tomto prípade si rozdelené podmnožiny uloží do objektov typu `string` (obrázok č. 23).

```
String poCiarku = stringPoCiarku.toString();
String odCiarky = stringOdCiarky.toString();
```

Obrázok č. 23: Ukážka zdrojového kódu z metódy algoritmus_Ekv_Kong, uloženie StringBufferov do stringov

Ak máme podmnožinu zo zadania rozdelenú na dve podmnožiny, algoritmus dvakrát zavolá metódu `kontrolujDvojprvky` (v týchto prípadoch nastaví oba vstupné parametre). Ako vidno na obrázku č. 24, pri prvom volaní metódy algoritmus nastaví ako prvý vstupný parameter podmnožinu, ktorá sa nachádzala po lomeno (reťazec `poCiarku` na obrázku č. 23). Ako druhý vstupný parameter nastaví podmnožinu za lomenom (reťazec `odCiarky` na obrázku č.23). Výstup z metódy `kontrolujDvojprvky` vloží do reťazca `poCiarku`. Pri druhom volaní metódy algoritmus nastaví ako prvý vstupný parameter podmnožinu, ktorá vznikne zlepením podmnožiny `poCiarku` a podmnožiny `pom_dva`.

Nastavanie podmnožiny `pom_dva` bolo priblížené pri rozoberaní metódy `kontrolujDvojprvky` v predchádzajúcej podkapitole.

Ako druhým vstupným parametrom je podmnožina `poCiarku`. Výstup z metódy vloží do reťazca `odCiarky`. Ako vidno na obrázku č. 24, tieto dve volania metódy sú vykonávané

v cykle, pokiaľ sa reťazec `pom_dva` nezhoduje s reťazcom `poCiarku`. Každou jednou iteráciou cyklu sa podmnožina `poCiarku` rozšíri o podmnožinu `pom_dva`.

```
int pomocna = 0;
while(pomocna == 0) {

    poCiarku = kontrolujDvojprvky(poCiarku, odCiarky);
    odCiarky = kontrolujDvojprvky(hlavneOknoData.spojPrvky(odCiarky, pom_dva), poCiarku);

    if(pom_dva.equals(poCiarku)) {
        pomocna = 1;
    } else {
        poCiarku = hlavneOknoData.spojPrvky(poCiarku, pom_dva);
    }
}
```

Obrázok č. 24: Ukážka zdrojového kódu z metódy algoritmus_ekv_kong, volanie metódy kontrolujDvojprvky

Po vyskočení z cyklu má algoritmus nájdené dve podmnožiny. Teraz musí rozhodnúť, či ponechá medzi nimi lomeno, alebo ich zlepí dokopy a lomeno vynechá. Ak sa v podmnožinách nachádza aspoň jeden spoločný prvok, znamená to, že ich môže zlepiť a odstrániť duplicitné prvky. Zlepenú podmnožinu prehlási za uzáver podmnožiny. Ak podmnožiny nemajú spoločný prvok, vloží medzi ne lomeno a výslednú podmnožinu prehlási za uzáver podmnožiny.

2.1.3 Metóda neporovnatelnePrvky

Metóda `neporovnatelnePrvky` prešla najviac rozšíreniami. V triede `Algoritmus_GZZN` sa spomínaná metóda skladá z dvoch cyklov a dvoch jednoduchých podmienok. V cykloch sa postupne lepia nájdené uzávery podmnožinám. Zlepené uzávery sa následne kontrolujú s už existujúcimi podmnožinami v príklade. Ak vznikne nová podmnožina, algoritmus k nej nájde uzáver.

V triede `Algoritmus_Ekv_Kong` je princíp rovnaký. Aj v tejto triede sa postupne lepia nájdené uzávery jednotlivých podmnožín. Pri týchto uzáveroch pred ich lepením treba kontrolovať, či sa v nich nenachádza lomeno. Ako vidno na obrázku č. 25, môžu nastať 4 možnosti:

- lomená v oboch uzáveroch,
- lomeno v prvom uzáveri,
- lomeno v druhom uzáveri,
- uzávery bez lomena.

```
for (int i = 0; i < LPTabulkaVygenerovana.size(); i++) {
    ParPodmnozin prvok_i = LPTabulkaVygenerovana.get(i);
    for(int j = i + 1; j < LPTabulkaVygenerovana.size(); j++) {

        if(prvok_i.getPravaStrana().contains("/")) {
            if(LPTabulkaVygenerovana.get(j).getPravaStrana().contains("/")) {
                //TODO lomeno v oboch uzáveroch
            } else {
                //TODO lomeno v prvom uzávere
            }
        } else {
            if(LPTabulkaVygenerovana.get(j).getPravaStrana().contains("/")) {
                //TODO lomeno v druhom uzávere
            } else {
                //TODO uzávery bez lomena
            }
        }
    }
}
```

Obrázok č. 25: Ukážka zdrojového kódu z metódy neporovnatelnePrvky, 4 možnosti pri lepení uzáverov

Ako prvú si priblížime situáciu, kedy oba lepené uzávery neobsahujú lomeno. V takomto prípade metóda musí najprv zistiť, či uzávery majú spoločný aspoň jeden prvok.

Ak uzávery majú spoločný prvok, metóda ich zlepí a vynechá duplicitné prvky. Následne skontroluje či sa zlepená podmnožina nachádza v LP tabuľke vygenerovanej na ľavej strane. Ak sa podmnožina v zozname nachádza, metóda ju už nebude druhýkrát pridávať. Ak sa nenachádza, tak pomocou metódy `kontrolujDvojprvky` jej nájde uzáver. Ak sa nájdený uzáver v LP tabuľke vygenerovanej ešte nenachádza, metóda túto dvojicu pridá do zoznamu. Ako vidno na obrázku č. 26, pred pridaním dvojice do zoznamu sa skontroluje veľkosť uzáveru s počtom prvkov vo zväze. Pridané sú len také dvojice, ktorých uzáver je menší ako počet prvkov vo zväze.

```
L_strana = hlavneOknoData.spojPrvky(prvok_i.getPravaStrana(), LPTabulkaVygenerovana.get(j).getPravaStrana());

if(hlavneOknoData.kontrolaZoznamuLava(L_strana) == 0) {

    P_strana = kontrolujDvojprvky(L_strana, "");

    if(P_strana.length() != hlavneOknoData.getVelkostZvazu() && (kontrolaZoznamuPrava(P_strana) == 0)) {
        LPTabulkaVygenerovana.add(new ParPodmnozin(L_strana, P_strana));
        hlavneOknoData.pridajRiadokPravo();
    }
}
```

Obrázok č. 26: Ukážka zdrojového kódu z metódy neporovnatelnePrvky, uzávery bez lomena, časť 1.

Ak uzávery nemajú spoločný prvok, metóda medzi ne vloží lomeno. Program dbá na to, aby prvky v jednotlivých podmnožinách boli usporiadané v abecednom poradí. Preto metóda musí rozhodnúť, ktorú podmnožinu vloží pred lomeno, a ktorú za lomeno. Na obrázku č. 27 vidíme, že po zostavení podmnožiny metóda znovu prekontroluje, či sa

nenachádza vo vygenerovanej LP tabuľke. Ak sa nenachádza, tak jej nájde uzáver. Na hľadanie uzáverov podmnožinám, ktoré obsahujú lomeno, slúži metóda `najdiUzaverPrvkomSLomitkom`. Metóda predpokladá na vstupe 3 parametre. Prvým je podmnožina, ku ktorej bude metóda hľadať uzáver. Druhým a tretím parametrom je podmnožina rozdelená podľa lomena. Funkcionalitu metódy si popíšeme neskôr.

```
if(prvok_i.getPravaStrana().charAt(0) < LPTabulkaVygenerovana.get(j).getPravaStrana().charAt(0)) {
    L_strana = prvok_i.getPravaStrana()+"/"+LPTabulkaVygenerovana.get(j).getPravaStrana();
} else {
    L_strana = LPTabulkaVygenerovana.get(j).getPravaStrana()+"/"+prvok_i.getPravaStrana();
}

if(hlavneOknoData.kontrolaZoznamuLava(L_strana) == 0) {
    najdiUzaverPrvkomSLomitkom(L_strana, prvok_i.getPravaStrana(), LPTabulkaVygenerovana.get(j).getPravaStrana());
}
```

Obrázok č. 27: Ukážka zdrojového kódu z metódy neporovnatelnePrvky, uzávery bez lomena časť 2.

Ďalej si priblížime situácie: keď sa v jednom uzávere lomeno nachádza a v druhom nie. Ako prvé je potrebné rozdeliť uzáver obsahujúci lomeno na dve podmnožiny. Tento postup je znázornený na obrázku č. 19. Teraz sa v metóde nachádza jeden uzáver bez lomena a dve podmnožiny druhého uzáveru, ktorý obsahoval lomeno. Ďalej metóda skontroluje, či vzniknuté podmnožiny druhého uzáveru nemajú aspoň jeden spoločný prvok s prvým uzáverom. Ak podmnožina má spoločný prvok s prvým uzáverom, metóda ju rozšíri o prvý uzáver. Môže sa stať, že rozšírené podmnožiny budú obsahovať spoločný prvok. Ak obsahujú takýto prvok, metóda ich zlepí dokopy. Takto vytvorí ľavú stranu novej podmnožiny. Ak sa podmnožina vo vygenerovanej LP tabuľke ešte nenachádza, zavolá sa metóda `najdiUzaverPrvkomSLomitkom`, ktorá nájde ľavej strane uzáver.

Ako poslednú si priblížime situáciu, kedy oba lepené uzávery obsahujú lomeno. Ako bolo zvykom aj v predchádzajúcich prípadoch, je nutné uzávery rozdeliť podľa lomena. V tomto prípade sa budú v programe nachádzať 4 podmnožiny z dvoch uzáverov obsahujúcich lomeno. Podmnožiny budú umiestnené do objektov typu StringBuffer, znázornených na obrázku č. 28.

```
final StringBuffer stringPoCiarku = new StringBuffer();
final StringBuffer stringOdCiarky = new StringBuffer();

final StringBuffer stringPoCiarku2 = new StringBuffer();
final StringBuffer stringOdCiarky2 = new StringBuffer();
```

Obrázok č. 28: Ukážka zdrojového kódu z metódy neporovnatelnePrvky, pomocné StringBuffre

Prvé dva sú určené pre prvý uzáver a druhé dva pre druhý uzáver. Po rozdelení si metóda prvú časť uzáveru uloží do reťazca `uzavarPoCiarku`, do ktorého začne zostavovať podmnožinu na

ľavej strane vygenerovanej LP tabuľky. Ako vidno na obrázku č. 29, postupne sa kontrolujú podmnožiny z druhého uzáveru s prvou časťou prvého uzáveru. Ak metóda nájde zhodu v prvku, rozšíri reťazec o podmnožinu. Tento postup opakuje, až pokiaľ sa reťazec už nerozširuje.

```
String uzaverPoCiarku = stringPoCiarku.toString();
String pomocna = "";
while(!pomocna.equals(uzaverPoCiarku)) {

    pomocna = uzaverPoCiarku;
    boolean nasolPoCarku = false;
    boolean nasolOdCarky = false;

    for(int a = 0; a < uzaverPoCiarku.length(); a++) {
        for(int b = 0; b < stringPoCiarku2.length(); b++) {
            if(uzaverPoCiarku.charAt(a) == stringPoCiarku2.charAt(b)) {
                nasolPoCarku = true;
            }
        }
        for(int b = 0; b < stringOdCiarky2.length(); b++) {
            if(uzaverPoCiarku.charAt(a) == stringOdCiarky2.charAt(b)) {
                nasolOdCarky = true;
            }
        }
    }

    if(nasolPoCarku) {
        uzaverPoCiarku = hlavneOknoData.spojPrvky(uzaverPoCiarku, stringPoCiarku2.toString());
    }
    if(nasolOdCarky) {
        uzaverPoCiarku = hlavneOknoData.spojPrvky(uzaverPoCiarku, stringOdCiarky2.toString() );
    }
}
```

Obrázok č. 29: Ukážka zdrojového kódu z metódy neporovnatelnePrvky, hľadanie podmnožiny po čiarku

Rovnakým spôsobom sa prekontroluje aj druhá podmnožina prvého uzáveru (`stringOdCiarky`). Podmnožinu si metóda uloží do reťazca `uzaverOdCiarky` a vykoná rovnaký postup, aký je znázornený na obrázku č. 29. Následne skontroluje, či reťazec `uzaverOdCiarky` a `uzaverPoCiarku` obsahujú rovnaký prvok. Ak áno, spoji ich. Ak nie, vloží medzi ne lomeno. Nakoniec zavolá metódu `najdiUzaverPrvkomSLomitkom`, ktorá nájde uzáver podmnožine.

2.1.4 Metóda najdiUzaverPrvkomSLomitkom

Metóda predpokladá na vstupe tri parametre. Prvým je podmnožina, ku ktorej bude hľadať uzáver. Druhým vstupným parametrom je prvá časť podmnožiny (po lomeno). Tretím parametrom je druhá časť podmnožiny (od lomena). Druhý a tretí vstupný parameter si uloží do pomocných reťazcov. Následne sa vykoná časť kódu, ktorú som už popisoval v metóde `kontrolujDvojprvky`, a ktorá je znázornená na obrázku č. 24. Na obrázku sa nachádza cyklus while, v ktorom sa do pomocných reťazcov nájdu podmnožiny, ktoré sú uzáverom

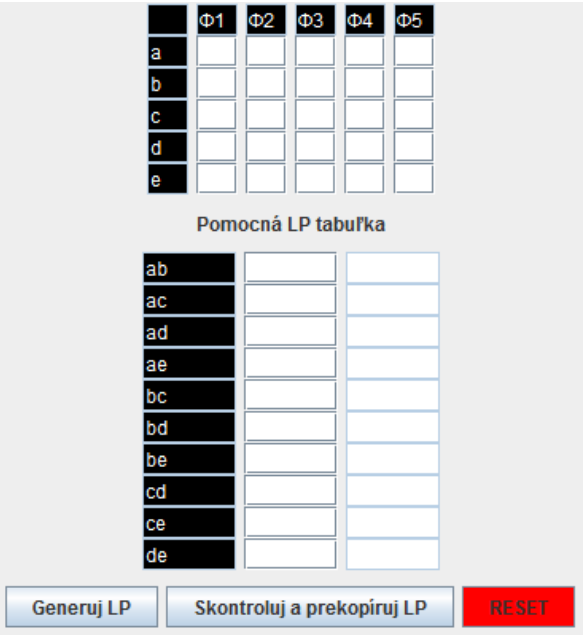
vstupných podmnožín (druhý a tretí vstupný parameter). Následne sa skontrolujú tieto uzávery či obsahujú spoločný prvok. Ak áno, metóda ich zlepí. Ak nie, vloží medzi ne lomeno. Teraz sa v metóde nachádza vstupná podmnožina aj s uzáverom. Ak sa tento uzáver ešte nenachádza vo vygenerovanej LP tabuľke, metóda ho tam vloží aj s jeho ľavou stranou (prvý vstupný parameter).

2.2 Trieda UnarneBinarneOperacie

Metódy z triedy `UnarneBinarneOperacie` sa využívajú len pri téme kongruencie. Umožňujú z unárnych/binárnych operácií v zadaní príkladu vygenerovať úvodnú LP tabuľku zadania príkladu, s ktorou sa pracuje pri predchádzajúcich témach. Vďaka tomu môže program pracovať univerzálne.

Generovanie tabuľky sa deje v novom okne. Je to hlavne z dôvodu prehľadnosti, aby boli oddelene kroky výpočtu, ktoré sa dejú nad rámec v téme kongruencie. V konštruktore triedy sa podľa operácii vygeneruje príslušné okno.

Tabuľku pre unárne operácie vytvorí metóda `tabulkaUnarne`, ktorá vráti komponent s tabuľkou, ktorá sa nachádza v hornej časti obrázku č. 30.



Obrázok č. 30: Okno pre zadanie príkladu s 5 unárnymi operáciami

Metóda pozostáva z dvoch cyklov. Ako vidno na obrázku č. 31, prvý riadok a prvý stĺpec v tabuľke je vyplnený, a nemožno do neho zapisovať. Pre vstupné polia v tabuľke, ktoré nie sú vyplnené, treba nastaviť, aké znaky sa do neho môžu zapisovať.

```
for(int i = 0; i < pocetprvkov; i ++) {
    for(int j = 0; j < pocetprvkov; j++) {
        JTextField field;
        if((i == 0)) {
            field = new JTextField(2);
            field.setBackground(Color.BLACK);
            field.setDisabledTextColor(Color.WHITE);
            field.setEnabled(false);
            if(j != 0) {
                znakJ++;
                stringBuffer.append((char) znakJ);
                field.setText(stringBuffer.toString());
                stringBuffer.deleteCharAt(0);
            }
        } else {
            if(j == 0) {
                field = new JTextField(2);
                field.setBackground(Color.BLACK);
                field.setDisabledTextColor(Color.WHITE);
                field.setEnabled(false);
                if(i == 0) {
                    field.setBackground(Color.BLACK);
                }
                field.setText("Φ"+i);

            } else {
                field = new JTextField(2);
            }
        }
        unarnePole[i][j] = nastavVstup(field,i,j,false,false);
    }
}
```

Obrázok č. 31: Ukážka zdrojového kódu z metódy tabulkaUnarne, vytvorenie tabuľky

Nastavenie pre vstupné polia má na starosti metóda `nastavVstup`. Ako vidno na obrázku č. 31, do metódy vstupuje päť parametrov. Prvým parametrom je vstupné pole, ktorému sa ide obmedziť vstup. Druhý a tretí parameter určujú polohu v dvojrozmernom poli. Štvrtý parameter určuje, či ide o tabuľku unárnych, alebo binárnych operácii. Piaty parameter určuje, či ide o pomocnú LP tabuľku. Podľa parametrov metóda presne vie, s ktorým vstupným polom, v ktorej tabuľke pracuje. Vďaka tomu vie presne obmedziť vstupujúce prvky. Pod tabuľkou unárnych operácii sa nachádza pomocná LP tabuľka. Tabuľku môžeme vidieť na obrázku č. 30 v strednej časti. Na vytvorenie tabuľky sa volá metóda `pridajTabulku`. Metóda v dvoch cykloch podľa počtu prvkov v konečnej množine vytvorí pomocnú LP tabuľku a predvyplní ju dvojprvkovými podmnožinami. Pod tabuľku sa nachádzajú tlačidlá na prácu s unárnymi/binárnymi operáciami a pomocnou LP tabuľkou.

Funkcionalita jednotlivých tlačidiel aj správny postup pri výpočte je sčasti prebraný v ďalšej kapitole a podrobne v používateľskej príručke v prílohe B – používateľská príručka.

V tabuľke sa nachádzajú dva P stĺpce. Do prvého stĺpca má používateľ možnosť postupne generovať zadanie príkladu. Po stlačení tlačidla *Generuj LP* sa v druhom stĺpci zobrazí správne zostavené zadanie z unárnych/binárnych operácií. Správne vygenerované

zadanie má na starosti metóda `dopl_LP_pom_unarne`.

Táto metóda postupne prechádza predvyplnené dvojprvkové podmnožiny v pomocnej LP tabuľke. Každú podmnožinu si rozdelí na prvky a tie následne hľadá v tabuľke unárnych operácii. Metóda si uloží indexy riadkov, v ktorých našla prvky z podmnožiny. V druhom kroku postupne k sebe lepí prvky v týchto dvoch riadkoch. Jednotlivé dvojprvkové podmnožiny oddelí bodkou. Takto pripraví reťazec pre metódu `orezKongruencie`. Táto metóda si reťazec prekopíruje do objektu triedy `StringBaffer` (`stringB`), aby sa s ním lepšie pracovalo. Postupne ho prechádza a odstraňuje z neho rovnaké podmnožiny. Po odstránení zbytočných podmnožín ho pošle do metódy `zlepKongruencie`. Metóda vráti P stranu LP tabuľky zadania.

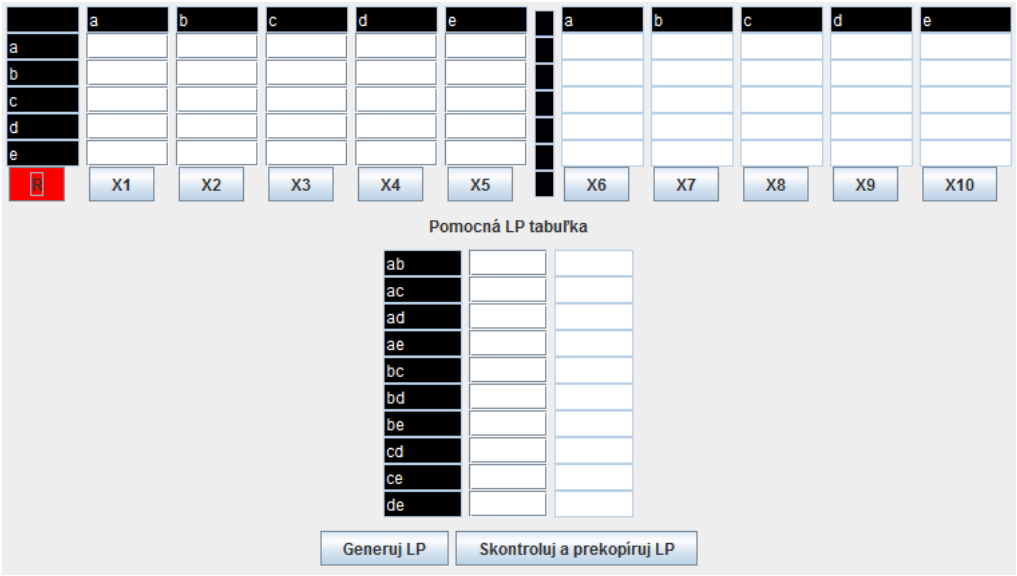
Metóda pri vytváraní podmnožiny do tabuľky postupuje v niekoľkých krokoch. V prvom kroku si prekopíruje prvú podmnožinu do pomocného reťazca (`pomocny_string`) a vymaže tri znaky z objektu `stringB`. V prvých dvoch znakoch bola uložená podmnožina a v tretom znaku oddeľovač podmnožín, bodka. V ďalšom kroku metóda hľadá zhodné prvky v jednotlivých podmnožinách v objekte `stringB` a v pomocnom reťazci. Ak nájde podmnožinu zhodnú aspoň v jednom prvku, odstráni ju aj s oddeľovačom a rozšíri o ňu pomocný reťazec. Ak v podmnožine nenájde zhodu, tak ju vynechá. Postup je znázornený na obrázku č. 32.

Keď prejde celý `stringB`, skontroluje, či je prázdny. Ak je prázdny, výsledná P strana LP tabuľky zadania sa nachádza v pomocnom reťazci. Ak nie je prázdny, pomocný reťazec si uloží do reťazca `poCiarku`. Do pomocného reťazca si opäť uloží prvú podmnožinu z objektu `stringB`. Podmnožinu aj s oddeľovačom následne odstráni z objektu. Teraz musí opäť prejsť celý `stringB`. Postupuje rovnako ako v predchádzajúcom prípade. Ak sa dostane na koniec, `stringB` je prázdny a v reťazcoch `pomocny_string` a `poCiarku` sa nachádzajú dve časti uzáveru podmnožiny. Tieto reťazce pošle do metódy `upracPodmnožiny`, ktorá ich spojí pomocou lomena a v abecednom poradí ich vráti.

```
while(stringBuffer.length() > a) {
    if(pomocny_string.contains(Character.toString(stringBuffer.charAt(a)))) {
        pomocny_string = this.spojPrvky(pomocny_string,
                Character.toString(stringBuffer.charAt(a))+Character.toString(stringBuffer.charAt(a+1)));
        stringBuffer.deleteCharAt(a);
        stringBuffer.deleteCharAt(a);
        stringBuffer.deleteCharAt(a);
        pomocna = 0;
    } else {
        if(pomocny_string.contains(Character.toString(stringBuffer.charAt(a+1)))) {
            pomocny_string = this.spojPrvky(pomocny_string,
                    Character.toString(stringBuffer.charAt(a))+Character.toString(stringBuffer.charAt(a+1)));
            stringBuffer.deleteCharAt(a);
            stringBuffer.deleteCharAt(a);
            stringBuffer.deleteCharAt(a);
            pomocna = 0;
        } else {
            a = a + 3;
        }
    }
}
```

Obrázok č. 32: Ukážka zdrojového kódu metódy zlepKongruencie, prechádzanie stringBaffru

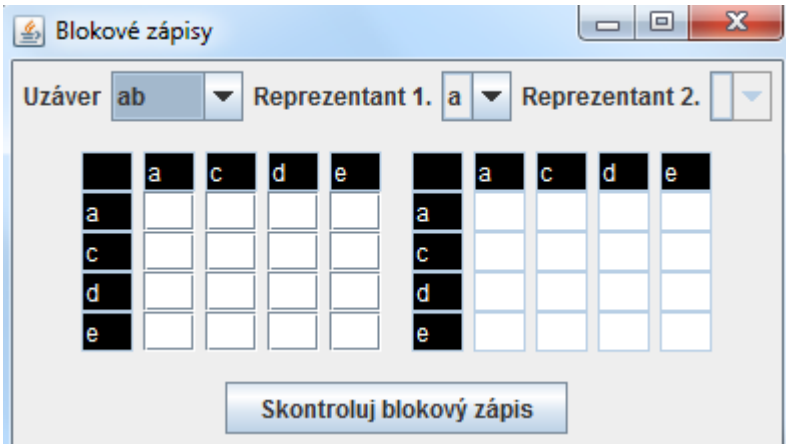
Tabuľka pre binárne operácie sa skladá z dvoch tabuliek určených pre operácie unárne. Jej vytvorenie má na starosti metóda `tabulkaBinarne`. Táto metóda vytvorí dve tabuľky, pričom je používateľovi dovolené písať len do prvej z nich. Druhá je určená len na čítanie. Druhá tabuľka sa vypĺňa z prvej tabuľky (program pretransformuje stĺpce na riadky). Ako vidno na obrázku č 33, tabuľky sú umiestnené v hornej časti vedľa seba. Je to z dôvodu prehľadnosti pri generovaní úvodnej LP tabuľky. Ako aj pri unárnych operáciách je zabezpečené, aby sa dali vpisovať do tabuliek len príslušné prvky zo zväzu. Generovanie úvodnej LP tabuľky pre binárne operácie sa deje podobným spôsobom ako pri unárnych operáciách. Pre binárne operácie sa volá metóda `dopl_LP_pom_binarne`. Rozdiel je v tom, že metóda okrem prvej tabuľky, ktorú používateľ vyplnil, prechádza aj druhú pretransformovanú tabuľku. V spodnej časti obrázka č. 33 sa opäť nachádza pomocná LP tabuľka.



Obrázok č. 33: Okno pre zadanie príkladu s 5 binárnymi operáciami

2.3 Trieda Blokove

Metódy z triedy `Blokove` sa využívajú pri hľadaní blokových zápisov kongruenciám. V parametrickom konštruktore sa nastavia základne informácie o príklade. Podľa týchto informácií si následne vytvorí okno určené na generovanie blokových zápisov.



Obrázok č. 34: Okno na generovanie blokových zápisov

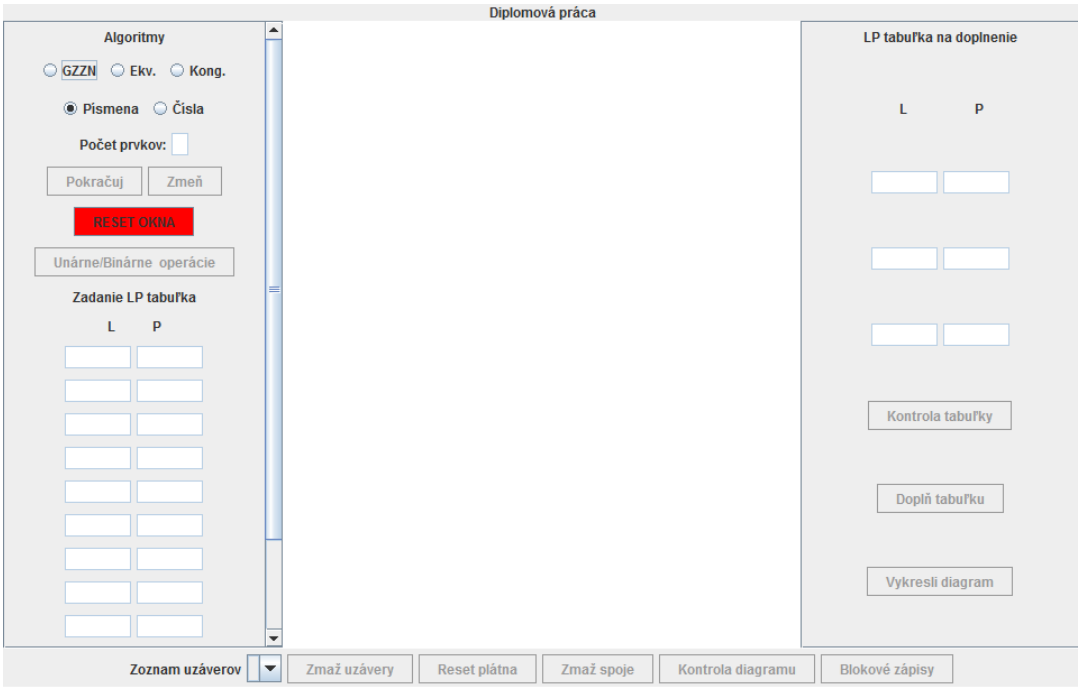
V okne sa nachádzajú všetky uzávery z plátna aj s možnosťou výberu reprezentantov. Zápisy sa kontrolujú s unárnymi, prípadne binárnymi operáciami zo zadania príkladu (Okno vytvorené triedou `UnarneBinarneOperacie`).

3 Pozadie, „zákulisie“ programu

Myšlienky v tejto kapitole už boli publikované v rámci študentskej vedeckej činnosti.

V tejto kapitole si sčasti priblížime užívateľské rozhranie programu, ale viacej sa budeme venovať „zákulisiu“ programu. Popíšeme si, čo sa v pozadí programu deje pri riešení príkladu. Zameriame sa na kroky výpočtu, ktoré sú pred používateľom skryté.

Pred zadaním zadania príkladu používateľ musí vyplniť základne informácie o príklade. Základné informácie sa skladajú z výberu algoritmu, spresnenia počtu prvkov vo zväze a typu prvkov, z ktorých sa zväz skladá (písmená, čísla). Tieto informácie sú dôležité pre celý výpočet. Ako vidno na obrázku č. 35, základné informácie sa nachádzajú na ľavej strane okna v hornej časti.



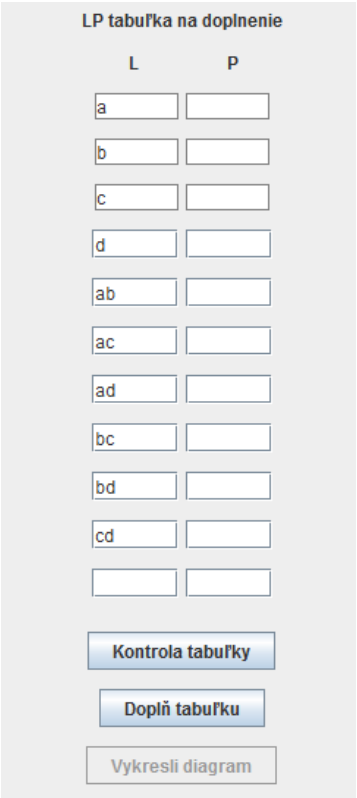
Obrázok č. 35: Hlavne okno programu horná časť

Obrázok č. 36: Hlavne okno programu spodná časť

Ak používateľ zvolil algoritmus kongruencie, je mu sprístupnené tlačidlo na

vygenerovanie nového okna (*Unárne/Binárne operácie*). V novom okne má možnosť zadať binárne alebo unárne operácie namiesto LP tabuľky zadania. Okno pre zadávanie binárnych operácií je znázornené na obrázku č. 33 a pre zadávanie unárnych operácií na obrázku č. 30. Z týchto operácií musí následne vygenerovať úvodnú LP tabuľku zadania. Algoritmus generovania tabuľky je podrobne popísaný v predchádzajúcej kapitole (2.2 Trieda UnarneBinarneOperacie). Program nedovolí používateľovi zavrieť pomocné okno s binárnymi/unárnymi operáciami. Je to z dôvodu neskoršieho hľadania blokových zápisov pre výsledné kongruencie. Vygenerovanú LP tabuľku v novom okne je možné prekopírovať do LP tabuľky zadania príkladu v hlavnom okne. (LP tabuľka na ľavej strane obrázku č. 35) Do tejto LP tabuľky sa pri témach generovanie zväzu zdola nahor a generovanie zväzov ekvivalencií hneď vpisuje zadanie príkladu.

Po vyplnení zadania príkladu, a stlačení tlačidla *Generuj*, sa v programe udeje pár dôležitých krokov. Ako prvé sa do ľavého stĺpca LP tabuľky na doplnenie predvyplnia podmnožiny z konečnej množiny. Pri témach generovanie zväzov ekvivalencií a kongruencií sú to len dvojprvkové podmnožiny. Pri téme generovanie zväzu zdola nahor ide o jedno a dvojprvkové podmnožiny z konečnej množiny. Na obrázku č. 37 sa nachádza LP tabuľka na doplnenie s predgenerovanými podmnožinami príkladu.



Obrázok č. 37: LP tabuľka na doplnenie s pred generovanými podmnožinami

Následne sa LP tabuľka zadania zablokuje a tabuľku nie je možné meniť. Ak však

používateľ zistí, že v zadaní je chyba, tabuľku môže znovu odomknúť na editáciu, prípadne celu premazať použitím tlačidiel *Späť* a *Zmaž zadanie*. Ako vidno na obrázku č. 36, okrem týchto dvoch tlačidiel sa pod LP tabuľkou zadania nachádza aj spomínane tlačidlo *Generuj*.

V ďalšom kroku program farebne odlíši správne vyplnené podmnožiny v príklade. Podmnožiny v LP tabuľke zadania, ktoré orámoval na zeleno, sú automaticky prekopírované do arrayListu `LPTabulkaZadania`. Tento arrayList reprezentuje LP tabuľku zadania v pozadí programu.

Je to z dôvodu, že s listom v pozadí sa lepšie pracuje ako s tabuľkou v hlavnom okne. Taktiež sa týmto prekopírovaním do pozadia oddelí časť programu, ktorá pracuje so zobrazovacími prvkami, a ktorá sa v pozadí venuje výpočtu.

Zároveň v pozadí programu vytvorí aj obraz LP tabuľky na doplnenie (arrayList `LPTabulkaVygenerovana`). V prvom kroku do neho naplní rovnaké podmnožiny, ako sú zobrazované v LP tabuľke na doplnenie (obrázok č. 37) . V druhom kroku im postupne nájde uzávery.

Pri hľadaní uzáverov sa nepozerá na LP tabuľku zadania, ktorá sa nachádza v okne, ale pracuje s arrayListom `LPTabulkaZadania`, ktorá túto tabuľku reprezentuje. Ako sa uzávery hľadajú som podrobne popisoval v predchádzajúcej kapitole, v ktorej som sa venoval aj jednotlivým metódam, ktoré boli pre tento účel vytvorené.

Nájdené uzávery je nutné ešte prekontrolovať. Program ich postupne lepí a zisťuje, či sa medzi nimi nenachádza taká kombinácia dvoch uzáverov, ktorých zlepením vznikne podmnožina, ktorej uzáver sa v arrayListe `LPTabulkaVygenerovana` ešte nenachádza. Ak takúto podmnožinu nájde, pridá ju aj s uzáverom do arrayListu a do LP tabuľky na doplnenie pridá prázdny riadok (obrázok č. 37).

Z toho vyplýva, že po stlačení tlačidla *Generuj* sa v pozadí programu nachádza výsledná LP tabuľka príkladu, reprezentovaná arrayListom `LPTabulkaVygenerovana`. Stlačením tlačidla *Doplň tabuľku* je možné pomocou tohto arrayListu doplniť LP tabuľku na doplnenie (obrázok č. 37). Ako vidno na obrázku č. 37, pod LP tabuľkou sa nachádza aj tlačidlo *Kontrola tabuľky*. Stlačením tohto tlačidla sa skontroluje správnosť nájdených uzáverov.

Ak používateľ našiel správne uzávery aspoň vygenerovaným podmnožinám, program tieto uzávery prekopíruje do zoznamu uzáverov. V zozname uzáverov sa nachádzajú aktuálne nájdené uzávery, ktoré používateľ ešte nepridal do Hasseovej diagramu. Zoznam uzáverov sa nachádza v spodnej časti obrázka č. 36. Vedľa zoznamu sa nachádzajú tlačidlá, ktoré sú určené na prácu s diagramom.

Ak používateľ nájde uzávery všetkým podmnožinám, má možnosť stlačením tlačidla *Kontrola tabuľky* overiť správnosť svojho riešenia. Ak program vyhodnotí zoznam za správny, odomkne možnosť automatického vykreslenia diagramu na plátno. Túto funkcionalitu má na starosti tlačidlo *Vykresli diagram*, ktoré sa nachádza pod LP tabuľkou na doplnenie. V jednom kroku s odomknutím tlačidla na automatické vykreslenie sa v pozadí programu vygeneruje Hasseovej diagram. Diagram sa v pozadí programu generuje do dvoch arrayListov v dvoch krokoch. V prvom kroku pre uzávery vygeneruje súradnice umiestnenia na plátne (arrayList `uzaver_G`) a v druhom kroku spoje medzi jednotlivými uzávermi (arrayList `hrana_G`).

Pred generovaním súradníc jednotlivým uzáverom treba vzniknuté uzávery zanalyzovať. Ako prvé odlíši vrcholy diagramu a zvyšné uzávery zoradí do polí podľa počtu prvkov. Podľa počtu vyplnených polí si plátno (obrázok č. 35 v strede) rozdelí na riadky. V každom riadku sa budú nachádzať uzávery s rovnakým počtom prvkov. Ich súradnice rozpočíta automaticky podľa počtu prvkov v poli tak, aby mali medzi sebou približne rovnaké medzery. Po nastavení súradníc uzáverom začne generovať spoje medzi nimi.

Program pri téme generovanie zväzu zdola nahor začína generovať spoje od dvojprvkových uzáverov a pri zvyšných témach až od trojprvkových uzáverov. Jednoprvkové uzávery v téme generovanie zväzu zdola nahor a dvojprvkové uzávery vo zvyšných dvoch témach automaticky spojí s prázdnou množinou. Spoje k uzáveru sa generujú vždy smerom nadol a to tak, že program hľadá všetky uzávery, ktoré pokrýva.

Predstavme si, že máme trojprvkový uzáver BCD. Tento uzáver zatiaľ nie je spojený so žiadnym uzáverom. Ako prvé, program skontroluje riadok pod ním a hľadá uzávery, ktoré môže pokryť. V tomto prípade by to boli uzávery BC, BD a CD. Ak nájde niektorý z týchto uzáverov, vytvorí s ním spoj, a názov uzáveru pridá do pomocnej premennej. Avšak, program do pomocnej premennej môže pridávať len prvky z uzáverov, ktoré ešte neobsahuje. Predstavme si, že by program našiel uzávery BC a CD. Do pomocnej premennej by uložil ako prvý uzáver BC a z druhého uzáveru by uložil už len prvok D. Ak prekontroluje celý riadok, zistí, či sa pomocná premenná nezhoduje s uzáverom. V našom prípade by sa zhodovala. Pomocná premenná by sa premazala a program by hľadal spoje k ďalšiemu uzáveru. Môže sa stať, že program v riadku pod uzáverom nájde len uzáver BC. V takomto prípade program prekontroluje aj ďalší riadok. Ak nájde chýbajúci uzáver, ktorý môže pokryť, pripojí ho, rozšíri pomocnú premennú a prekontroluje, či sa pomocná premenná zhoduje s uzáverom, ku ktorému hľadá spoje. Ak áno, hľadanie spojov k danému uzáveru ukončí a ak nie, pokračuje ďalším riadkom. Ak sa program prepracuje až k poslednému riadku a nemá pokryté všetky

uzávery, ktoré by mohol mať, skontroluje, či sa mu podarilo pokryť aspoň jeden. Ak pomocná premenná nie je prázdna, znamená to, že pokryl aspoň jeden uzáver. V takomto prípade program už nevytvára žiadne spoje k tomuto uzáveru a pokračuje v hľadaní spojov k ďalšiemu uzáveru. Ak je však prázdna, znamená to, že uzáver nebol spojený so žiadnym iným uzáverom, a preto ho program spojí s prázdnou množinou. Tým zamedzí tomu, že by sa v Haseeovej diagrame nachádzali uzávery, ktoré nič nepokrývajú. Týmto spôsobom pospája uzávery s uzávermi, ktoré pokrývajú. Nakoniec musí prekontrolovať, či je každý uzáver spojený s niektorým uzáverom nad ním. Ak nájde uzáver, ktorý ma spoje len smerom nadol, spojí ho s uzáverom celej množiny. Týmto sa zamedzí, aby sa v Hasseovej diagrame nenachádzali uzávery, ktoré by len „viseli“ vo vzduchu. Takto vygenerovaný diagram je možné stlačením tlačidla *Vykresli diagram* vykresliť na plátno. Okrem toho, tento diagram slúži aj na kontrolu. Kliknutím na tlačidlo *Kontrola diagramu* program zistí, či nakreslený diagram je zhodný s diagramom vygenerovaným, a ak nie je, upozorní na chyby.

Pri téme kongruencie sa po nakreslení diagramu musia nájsť blokové zápisy všetkým uzáverom v diagrame. Po správnom nakreslení diagramu sa aktivuje tlačidlo *Blokové zápisy*, ktoré umožní používateľovi v novom okne na obrázku č. 34 tieto zápisy generovať. V okne sa nachádzajú všetky uzávery z plátna aj s možnosťou výberu reprezentantov. Zápisy sa kontrolujú s unárnymi (obrázok č. 30), prípadne binárnymi operáciami (obrázok č. 33), ktoré používateľ zadal na začiatku príkladu.

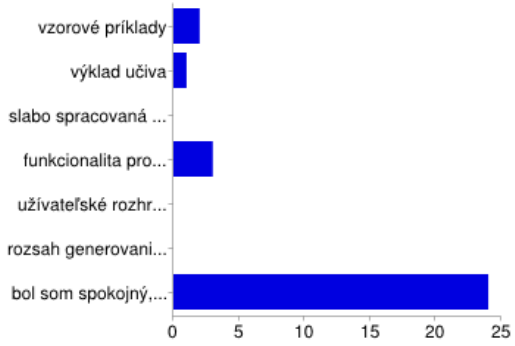
Funkcionalita všetkých tlačidiel je dopodrobna rozobraná v užívateľskej príručke, ktorá sa nachádza v prílohe B – užívateľská príručka. Okrem funkcionality tlačidiel a popisu užívateľského rozhrania sa v príručke nachádza aj krátka zbierka príkladov ku každej téme.

4 Využitie programu

V tejto časti práce si zhrnieme a vyhodnotíme spätnú väzbu od testovacej skupiny, ktorej bol program poskytnutý. Testovacej skupine bola okrem programu poskytnutá aj užívateľská príručka a krátky dotazník. Testovacia skupina sa skladala z 30tich študentov fakulty, ktorí sa stretli s predmetom diskrétna matematika. V dotazníku sa nachádza 12 jednoduchých otázok, v ktorých mal študent možnosť vyjadriť svoj názor na program a jednotlivé témy. V otázkach týkajúcich sa spokojnosti a nespokojnosti s programom alebo témou mal študent možnosť označiť viacej možností a tým lepšie vyjadriť svoj názor. Z toho vypláva, že pri každej odpovede mohlo byť dosiahnutých 0 až 100%. Dotazník sa nachádza v prílohe A - dotazník.

4.1 Hodnotenie témy generovanie zväzu zdola nahor

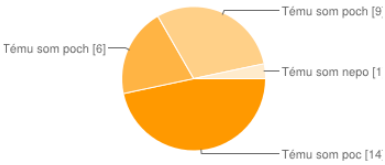
Pri pohľade na graf číslo 1 si môžeme všimnúť, že až 80% študentov bolo so spracovaním témy spokojných a skoro nič im nechýbalo. Študentom sa najviac páčilo užívateľské rozhranie a rozsah generovania zväzov. Rozsah generovania zväzov je pri tejto téme najväčší, až osem prvkov.



vzorové príklady	2	7 %
výklad učiva	1	3 %
slabo spracovaná príručka	0	0 %
funkcionalita programu	3	10 %
užívateľské rozhranie programu	0	0 %
rozsah generovania zväzov	0	0 %
bol som spokojný, nič mi nechýbalo	24	80 %

Graf č. 1: GZZN - nedostatky spracovania témy

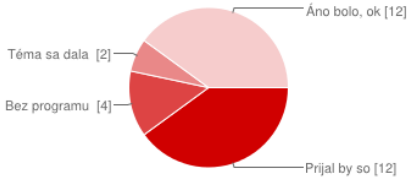
Graf číslo 2 nám znázorňuje percentuálne pochopenie témy medzi študentmi. Ako si môžeme všimnúť, až 47% študentov tému pochopilo už na prednáške a používaním programom sa zdokonalili v riešení príkladov. 20% študentov nepotrebovalo na pochopenie témy vytvorený program. Na druhej strane až 30% študentov pochopilo tému až vďaka programu. Jeden študent tému nepochopil vôbec.



Tému som pochopil už na prednáške. Používaním programu som sa zdokonalil v riešení príkladov.	14	47 %
Tému som pochopil už na prednáške, program bol pre mňa zbytočný.	6	20 %
Tému som pochopil vďaka programu.	9	30 %
Tému som nepochopil ani na prednáške, ani pri používaní programu. Prijal by som podrobnejšie spracovanie.	1	3 %

Graf č. 2: GZZN - pochopenie témy

Graf číslo 3 znázorňuje vyjadrenie študentov na otázku, či bolo nutné danú tému spracovávať. 40% študentov by prijalo spracovanie iných, zložitejších tém z diskrétnej matematiky. Zhodne sa tiež 40% študentov vyjadrilo, že bolo potrebné tému spracovávať a okrem programu by prijalo aj podrobnejšie skriptá s viacerými príkladmi. Z grafu tiež vyplýva, že 13% študentov by danú tému bez programu nepochopilo.



Prijal by som spracovanie iných, zložitejších tém.	12	40 %
Bez programu by som danú tému nepochopil.	4	13 %
Téma sa dala dobre pochopiť aj zo skrípt.	2	7 %
Áno bolo, okrem programu by som prijal aj podrobnejšie skriptá s viacerými príkladmi.	12	40 %

Graf č. 3: GZZN - výber témy

4.2 Hodnotenie témy ekvivalencie

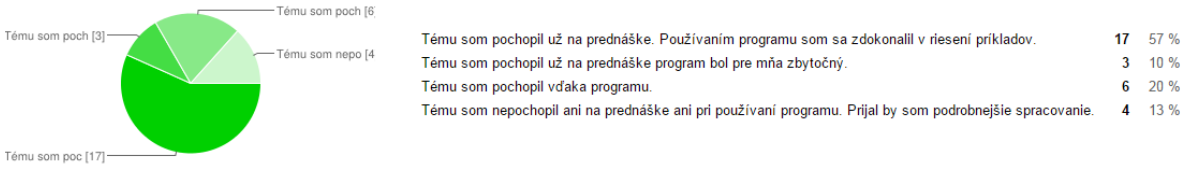
Na grafe číslo 4 vidno, že až 87% študentov bolo so spracovaním témy spokojných a nič im nechýbalo. Ako aj pri téme generovanie zväzu zdola nahor, aj pri téme ekvivalencií boli študenti najviac spokojní s užívateľským rozhraním programu.

vzorové príklady	1	3 %
výklad učiva	1	3 %
slabo spracovaná príručka	2	7 %
funkcionalita programu	1	3 %
užívateľské rozhranie programu	0	0 %
rozsah generovania zväzov	2	7 %
bol som spokojný, nič mi nechýbalo	26	87 %

Graf č. 4: Ekvivalencie - nedostatky spracovania témy

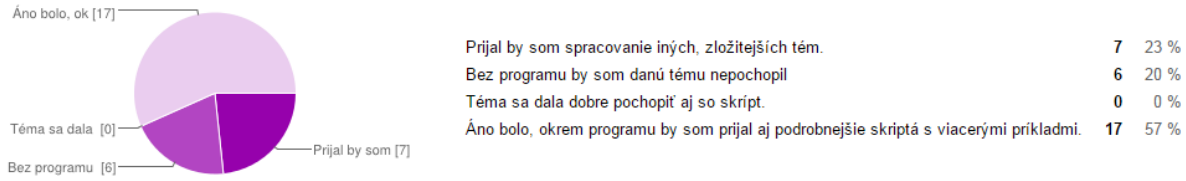
Na grafe číslo 5 vidno, že až 57% študentov tému pochopilo už na prednáške a používaním programu sa v danej téme zdokonalili. 10% študentov nepotrebovalo vytvorený program použiť. Na druhej strane až 20% študentov sa zhodlo, že tému by bez programu nepochopilo.

13% študentov danú tému nepochopilo.



Graf č. 5: Ekvivalencie - pochopenie témy

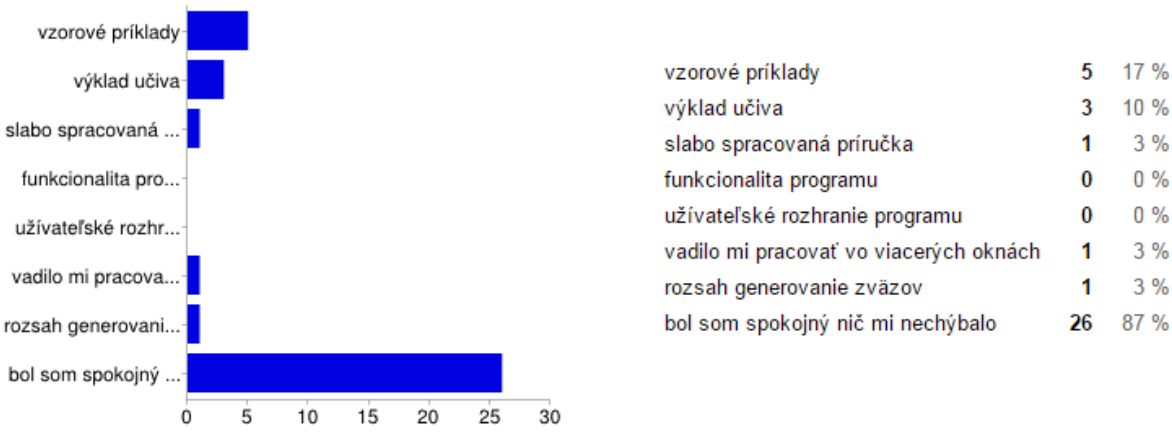
Na grafe číslo 6 si môžeme všimnúť, že 23% študentov by prijalo spracovanie iných, zložitejších tém. Naopak až 20% študentov by tému bez programu nepochopilo. Až 57% študentov sa zhodlo, že tému bolo nutné spracovať a prijali by podrobnejšie skriptá aj s príkladmi. Žiaden študent si nemyslí, že sa téma dala dobre pochopiť zo skrípt.



Graf č. 6: Ekvivalencie - výber témy

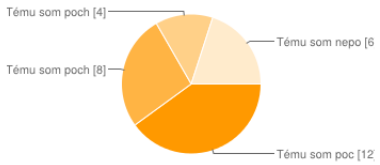
4.3 Hodnotenie témy kongruencie

Na grafe číslo 7 si môžeme všimnúť, že až 87% študentov bolo so spracovaním témy spokojných a nič im nechýbalo. Opäť z hodnotenia najlepšie vyšlo užívateľské rozhranie programu, s ktorým aj do tretice boli užívatelia najviac spokojní.



Graf č. 7: Kongruencie - nedostatky spracovania témy

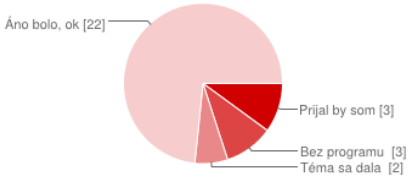
Na grafe číslo 8 si môžeme všimnúť, že 40% študentov pochopilo tému na prednáške a používaním programu sa zdokonalili v riešení príkladov. 27% študentov sa vyjadrilo, že tému pochopili na prednáške a program nepotrebovali. Na druhej strane, 13% študentov k pochopeniu témy potrebovalo program. Až 20% študentom nepomohol ani program k pochopeniu témy.



Tému som pochopil už na prednáške. Používaním programu som sa zdokonalil v riešení príkladov.	12	40 %
Tému som pochopil už na prednáške. program bol pre mňa zbytočný.	8	27 %
Tému som pochopil vďaka programu.	4	13 %
Tému som nepochopil ani na prednáške, ani pri používaní programu. Prijal by som podrobnejšie spracovanie.	6	20 %

Graf č. 8: Kongruencie - pochopenie témy

Na grafe číslo 9 jasne dominuje vyjadrenie, že danú tému bolo treba spracovať a až 73% študentov by prijalo rozsiahlejšie spracovanie aj s príkladmi. 10% študentov by danú tému bez programu nepochopilo.

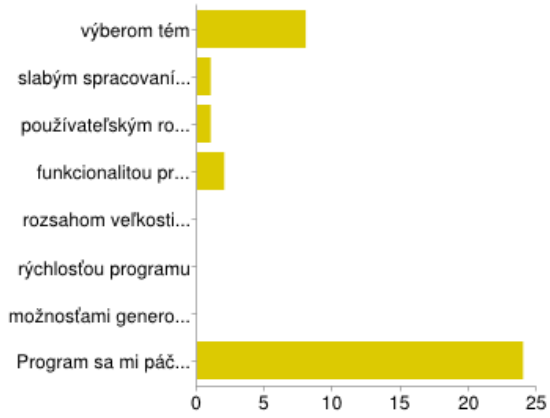


Prijal by som spracovanie iných, zložitejších tém.	3	10 %
Bez programu by som danú tému nepochopil.	3	10 %
Téma sa dala dobre pochopiť aj zo skript.	2	7 %
Áno bolo, okrem programu by som prijal aj podrobnejšie skriptá s viacerými príkladmi.	22	73 %

Graf č. 9: Kongruencie - výber témy

4.4 Hodnotenie programu celkovo

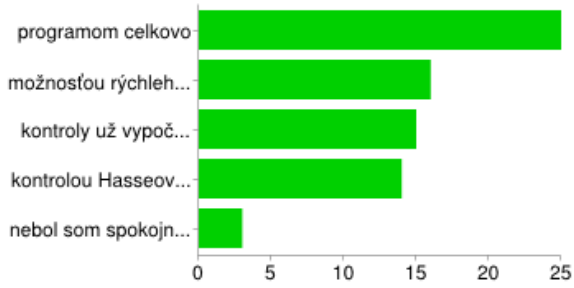
Na grafe číslo 10 sa nachádza celkové hodnotenie nespokojnosti s programom. 27% študentov bolo nespokojných s výberom tém. Na druhej strane až 80% študentov sa vyjadrilo, že sa im program páčil a boli s ním spokojní.



výberom tém	8	27 %
slabým spracovaním zvolených tém	1	3 %
používateľským rozhraním	1	3 %
funkcionalitou programu celkovo	2	7 %
rozsahom veľkosti zväzov	0	0 %
rýchlosťou programu	0	0 %
možnosťami generovania	0	0 %
Program sa mi páčil, bol som s ním spokojný.	24	80 %

Graf č. 10: Program - nespokojnosť

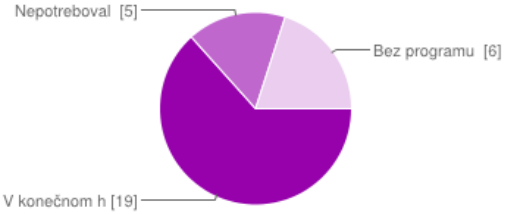
Na grafe číslo 11 vidíme zachytenú spokojnosť s programom. Až 83% študentov sa vyjadrilo, že boli s programom spokojní. 53% študentom sa páčilo rýchle generovanie zväzov. 50% študentov sa páčila kontrola vypočítaných príkladov a 47% sa vyjadrilo, že boli spokojní aj s kontrolu výsledného Hasseovej diagramu. Len 10% študentom sa v programe nič nepáčilo a neboli s ním spokojní.



programom celkovo	25	83 %
možnosťou rýchleho generovania príkladov	16	53 %
kontroly už vypočítaných príkladov	15	50 %
kontrolou Hasseových diagramov	14	47 %
nebol som spokojný s ničím	3	10 %

Graf č. 11: Program - spokojnosť

V záverečnom hodnotení programu, ktoré je znázornené na grafe číslo 12, sa 63% študentov vyjadrilo, že sa im program páčil a boli s ním spokojní. 17% študentov sa vyjadrilo, že program nepotrebovali použiť a témam porozumeli aj bez neho. Na druhej stane, 20% študentov sa vyjadrilo, že bez programu by malo problém porozumieť týmto trom témam.



V konečnom hodnotení mi program pomohol.	19	63 %
Nepotreboval som použiť tento program.	5	17 %
Bez programu by som mal problém pochopiť tieto tri témy.	6	20 %

Graf č. 12: Program – názor

4.5 Zhodnotenie spätnej väzby

Ponúknutý program mal dobrú odozvu medzi študentmi. Väčšina vyjadrila spokojnosť so spracovaním vybraných tém. Téma kongruencie, ktorú považujem za najťažšiu, mala najvyššie pozitívne hodnotenie, aj keď niektorým študentom chýbalo viac vzorových príkladov. Myslím, že skôr ako viac príkladov som mohol podrobnejšie ponúknuté príklady rozobrať. Napriek tomu sa hodnotenie témy blížilo až k 90%. Rovnako dobre dopadla aj téma ekvivalencie. Pri téme ekvivalencia bola spokojnosť s príkladmi väčšia, napriek tomu, že som postupoval pri tvorení vzorových príkladov rovnako, ako v téme kongruencie. Z toho vyplýva, že téma kongruencie bola pre študentov ťažšie pochopiteľná ako téma ekvivalencie. Avšak podľa môjho názoru boli príklady na obe témy dostatočne vysvetlené a dobre spracované. Pri téme generovanie zväzu zhora nadol ma prekvapilo vyjadrenie nespokojnosti študentov s funkcionalitou programu, napriek tomu, že funkcionalita programu je pri tejto téme najjednoduchšia a ľahko pochopiteľná.

Pochopenie jednotlivých tém programu prinieslo zaujímavé výsledky. Pri téme kongruencie boli jednotlivé odpovede najviac vyrovnané. Na jednej strane ani vďaka programu, ani vysvetleniu 20% študentov tému nepochopilo. Na strane druhej, 40% študentov

tému pochopilo na prednáške a používaním programu sa zdokonalili v riešení príkladov. Z týchto čísiel nám jasne vyplýva, že túto tému nepochopila väčšina študentov už na prednáške. Pätine študentov nepomohol k pochopeniu témy ani poskytnutý program. Tento výsledok ma prekvapil, keďže istá skupina študentov pochopila túto tému až vďaka programu. Z toho mi jasne vyplýva, že spracovanie danej témy nemohlo byť až na takej zlej úrovni, ako sa na prvý pohľad zdá. Pri téme ekvivalencie už nie sú odpovede také vyrovnané. Jasne dominuje vyjadrenie študentov, že vďaka programu sa zlepšili v riešení príkladov. Podobný názor mali študenti aj pri téme generovanie zväzu zdola nahor. Aspoň takýto výsledok som očakával pri každej téme, nakoľko program bol vytvorený práve na tento účel.

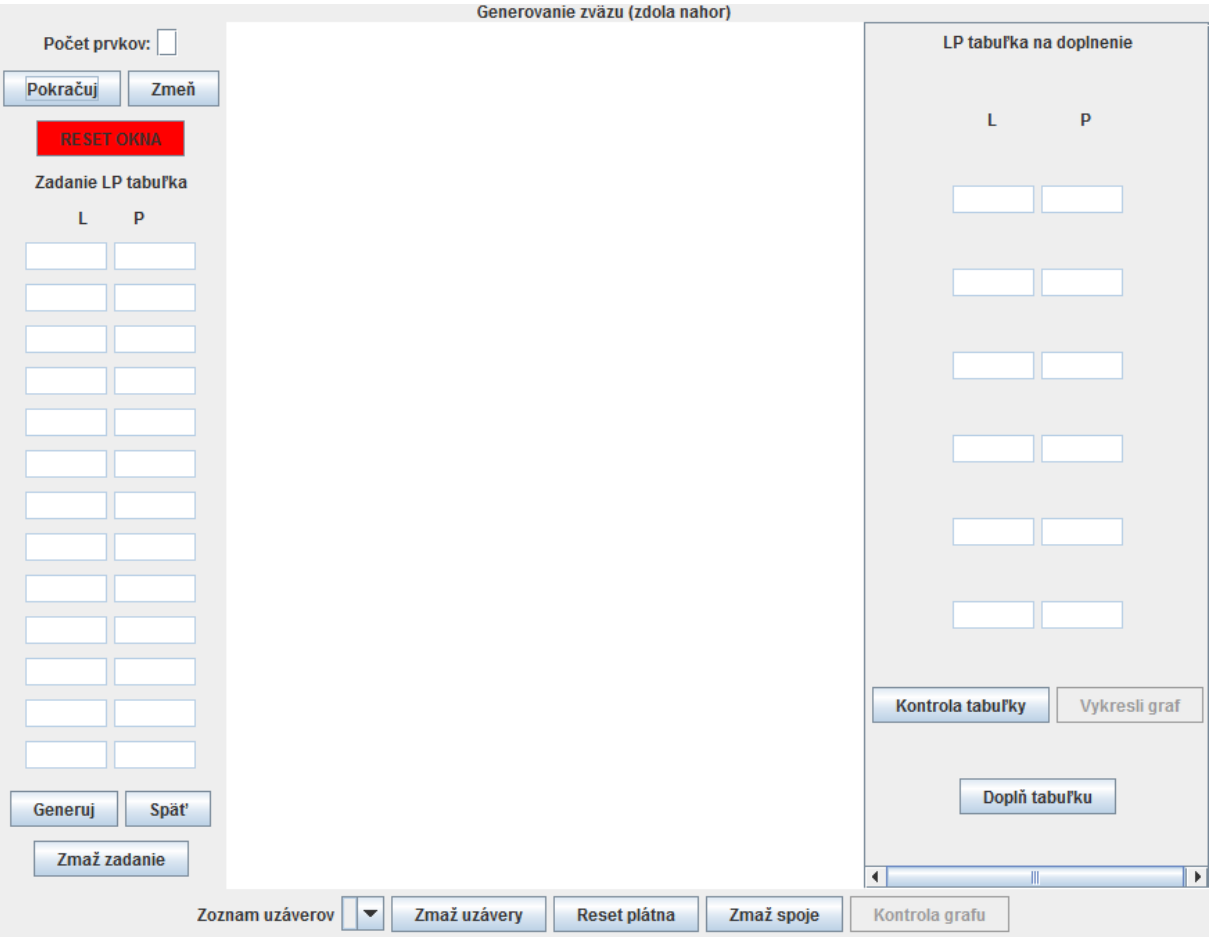
Pri vyjadrení názoru, či témy bolo nutné spracovávať, sú výsledky taktiež zaujímavé. Najväčší rozpor odpovedí je pri prvej téme (generovanie zväzu zdola nahor), kde sa zhodne 40% študentov vyjadrilo, že by prijalo spracovanie zložitejších tém, no na druhej strane by 40% študentov prijalo podrobnejšie skriptá s viacerými príkladmi na danú tému. Osobne si myslím, že spracovanie tejto témy bolo potrebné, keďže len 20% študentov sa vyjadrilo, že program bol pre nich zbytočný. Nadpolovica študentov v téme ekvivalencie a tri štvrtiny študentov v téme kongruencie je toho názoru, že okrem programu by prijali aj podrobnejšie skriptá a viac príkladov.

Pri záverečnom hodnotení programu sa skoro každému tretiemu študentovi nepáčil výber tém. Na druhej strane až 80% študentov sa vyjadrilo, že bolo s programom spokojných. Študenti nevyčlenili len určitú časť funkcionality programu, ale program sa im páčil ako celok. V konečnom hodnotení program študentom pomohol.

Aj keď sú názory študentov na vytvorený program rozporuplné, teší ma, že istej skupine program pomohol ako učebná pomôcka. Myslím si, že by bolo dobre, keby bol program k dispozícii študentom ako učebná pomôcka. Tvrdím to na základe dosiahnutých pozitívnych výsledkov. Pri používaní programu by sa výsledky dosiahnuté v týchto troch témach zhoršiť nemohli, naopak. Myslím, že by sa len zlepšili.

Záver

Pri rozširovaní teoretickej časti práce som sa nechal inšpirovať skriptami Doc. RNDr. Bohuslava Siváka, CSc. Pri rozširovaní programu to už také jednoduché nebolo. Aj keď na prispôsobenie hlavného okna pre algebry stačilo pridať len výber z algoritmov a dve tlačidlá. Kvôli pridaním prvkov bolo nutné upraviť rozloženie komponentov v okne. Na obrázku číslo 38 sa nachádza hlavné okno programu bakalárskej práce.



Obrázok č. 38: Ukážka hlavného okna z bakalárskej práce

Pri rozširovaní funkcionality bolo jasné, že bude nutné pridať nové triedy. V prvom kroku som pridal štyri nové triedy. Trieda `UnarneBinarneOperacie` vygeneruje nové okno, v ktorom má používateľ možnosť vytvoriť si úvodnú LP tabuľku zadania pomocou binárnych alebo unárnych operácií. Túto možnosť má len pri téme kongruencie. Druhou pridanou triedou bola trieda `Blokove`. Táto trieda sa opäť viaže ku téme kongruencie. Po správnom nakreslení Hasseovej diagramu má študent možnosť vygenerovať vzniknutým kongruenciám

blokové zápisy. Treťou pridanou triedou bola trieda `PomocnaLPTabulka`. Táto trieda reprezentuje jeden riadok v pomocnej LP tabuľke, ktorá sa využíva v triede `UnarneBinarneOperacie`. Štvrtou pridanou triedou bola trieda `VyberAlgoritmu`. V triede sa uchováva zoznam algoritmov, s ktorým program pracuje.

V druhom kroku som musel rozhodnúť, ako správne modifikovať už existujúce triedy. Najprv som sa pokúšal rozšíriť existujúce triedy tak, aby zvládli funkcionalitu vybraných tém. Podarilo sa mi to pri väčšine tried, až na triedu `Algoritmus`. Najviac práce bolo pri triedach `Graf`, `GenerujGraf` a `HlavneOknoFunkcie`. Do triedy `HlavneOknoFunkcie` bola pridaná skupina metód, ktoré sa volajú len pri témach kongruencie a ekvivalencie. Napríklad prístupnosť nových tlačidiel alebo úprava základných nastavení programu. Metódy v triedach `Graf` a `GenerujGraf` boli prispôsobené tak, aby pri témach ekvivalencie a kongruencie rátali s 3 až 5 prvkovým zväzom a dvoj a viac prvkovými podmnožinami. Pri téme Generovanie zväzu zdola nahor sa aj naďalej predpokladá s 3 až 8 prvkovými zväzmi a jedno a viac prvkovými podmnožinami tak, ako to bolo v bakalárskej práci. Pri automatickom generovaní spojov v Hasseovej diagrame bolo nutné rátať aj s uzávermi obsahujúcimi lomeno. Týmto sa generovanie diagramu skomplikovalo. Metódy v triede `GenerujGraf` sú však prispôsobené aj na takého uzávery.

Pri triede `Algoritmus` som sa rozhodol, že ju rozširovať nebudem, ale vytvorím novú triedu pre generovanie uzáverov pre zvyšné témy. Kvôli prehľadnosti som už existujúcu triedu `Algorimus` premenoval na triedu `Algoritmus_GZZN` a vytvoril novú triedu `Algoritmus_Ekv_Kong`. V tejto triede sa nachádza väčšina metód z triedy `Algoritmus_GZZN`, rozšírená o možnosť generovať uzávery obsahujúce lomeno.

Výsledný program bol ponúknutý študentom ako učebná pomôcka. Väčšina študentov sa vyjadrila, že sa im program páčil a pomohol im pri učení. To znamená, že ak by sa program používal v rámci výučby, pochopenie tém a s tým spojené výsledky by boli lepšie.

Zoznam bibliografických odkazov

[1] UŠÁK, Š.: *Demonštračný program pre diskrétnu matematiku*. Bakalárska práca Banská Bystrica: Univerzita Mateja Bela, 2013. 51s.

[2] SIVÁK, B.: *Diskrétna matematika*. 1.vyd. Banská Bystrica: Univerzita Mateja Bela, 2009. 142 s. ISBN 978-80-8083-800-3

[3] TOMAN, E.: *Úvod do diskrétnych štruktúr* [online], Dostupné na internete: <http://www.dcs.fmph.uniba.sk/studium/UvodDoDiskretnychStruktur/UDS-prednasky.pdf> [cit. 11.2.2014]

[4] Oracle :*Java Java™ Platform, Standard Edition 6 API Specification* [online], Dostupne na internete: <http://docs.oracle.com/javase/6/docs/api/index.html> [cit. 17.2.2014]

[5] Oracle. : *The Java™ Tutorials* [online], Dostupne na internete: < http://docs.oracle.com/javase/tutorial/> [cit. 22.2.2014]

[6] ZAKHOUR, S., a kolektív.: *Java 6 Výukový kurz.* 1.vyd. Brno: Computer Press, 2007. 536 s. ISBN 987-80-251-1575-6

[7] *Aplikovaná algebra* [online], Dostupne na internete: <http://mariokapusta.eu.sk/stahuj/algebra.pdf> [cit. 28.12.2014]

[8] *Discrete Mathematics Books* [online], Dostupne na internete: <http://www.freebookcentre.net/Mathematics/Discrete-Mathematics-Books.html> [cit. 3.12.2014]

[9] *Zbierka úloh z diskrétnej matematiky* [online], Dostupne na internete: < http://web.tuke.sk/fei-km/sites/default/files/prilohy/10/Zbierka-DM.pdf> [cit. 2.3.2015]

[10] KANISOVÁ, H., MLLER, M.: *UML srozumitelně*, 1.vyd. Brno: Computer Press, 2004. 158 s. ISBN 80-251-0231-9

[11] UŠÁK, Š., BRODENEC, I.: *Podporný program pre vyučovanie diskrétnej matematiky* In: *Prírodovedec 2014 Zborník príspevkov zo ŠVK 2014.* Banská Bystrica: Fakulta prírodných vied Univerzita Mateja Bela, 2014. 285 s. ISBN 978-80-557-0721-1