Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Aplikace pro výuku stromů

2015

Jakub Malíšek

Vedoucí práce: Mgr. Jiří Zacpal, Ph.D.

Studijní obor: Aplikovaná informatika, prezenční forma

Bibliografické údaje

Autor:	Jakub Malíšek
Název práce:	Aplikace pro výuku stromů
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2015
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Jiří Zacpal, Ph.D.
Počet stran:	40
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Jakub Malíšek
Title:	Application for teaching trees
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2015
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Jiří Zacpal, Ph.D.
Page count:	40
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Aplikace EducTree slouží jako podpora výuky datových struktur stromů. Teoretická část obsahuje kapitoly věnující se některým strukturám stromů. Praktická část tyto stromy reprezentuje v grafické podobě a umožňuje práci s nimi. Ke každému stromu jsou přiděleny testy k ověření znalostí uživatele. Aplikace byla vytvořena v jazyce Java jako součást bakalářské práce.

Synopsis

Application EducTree serves as a support for teaching data structures of trees. Teoretical part contains chapters dedicated to selected tree structures.Practical part represents theese trees in graphical form and allows working with them. Each tree has own tests to check knowlage of user. Application was created in Java language as a part of bachelor thesis.

Klíčová slova: datová struktura; stromy; AVL strom; červeno-černý strom; B-strom

Keywords: data structure; trees; AVL tree; red-black tree; B-tree

Děkuji Mgr. Jiřímu Zacpalovi, Ph.D za odborné vedení práce. Děkuji také Bc. Kateřině Kratochvílové za věcné připomínky a pomoc při testování aplikace.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam vět

Seznam zdrojových kódů

1 Úvod

Cílem bakalářské práce je vytvořit vzdělávací aplikaci určenou pro výuku datových struktur stromů. Stromy jsou důležité datové struktury s mnoha aplikacemi. Patří však ke složitějším datovým strukturám, které bývají obtížné na pochopení bez správné vizualizace. Aplikace by měla sloužit jak jako zdroj studijních materiálů, tak jako prostředek k vizualizaci zvolených stromů a testování znalostí uživatele. V následujících podkapitolách uvedeme specifikaci požadavků, průzkum existujících řešení a graf případu užití. Další kapitoly práce pak obsahují programátorskou a uživatelskou dokumentaci.

1.1 Specifikace požadavků

Výstupem práce bude aplikace sloužící jako podpora výuky datových struktur stromů. Konkrétně se bude jednat o stromy požadované u státní závěrečné zkoušky oborů Aplikovaná informatika[5] a Informatika[4] na Univerzitě Palackého. Tyto stromy jsou:

- Binární vyhledávací strom[1]
- AVL strom
- Červeno-černý strom
- B-strom

Aplikace by měla najít využití jak v samostudiu, tak jako podpora výuky pro předměty zabývající se datovými strukturami stromů. Bude rozdělena na tři části:

1. Teoretická část
2. Praktická část
3. Testová část

1.1.1 Teoretická část

Teoretická část by měla obsahovat úvod do problematiky stromů a dále kapitoly věnující se jednotlivým stromům a jejich struktuře. U každého stromu by pak měl být vysvětlen postup vyhledání, vložení a odebrání prvku. Teoretická část by měla být přístupná odkudkoliv z aplikace a mělo by v ní být umožněno hledání a kopírování textu.

[1] Dále v textu BST (zkratka z binary search tree)

1.1.2 Praktická část

V praktické části by měl být každý strom naprogramován a vizuálně ztvárněn. Uživatel by měl mít možnost provádět nad stromy operace vložení, odebrání a vyhledání prvku. Aplikace by měla obsahovat okno s textem, ve kterém by se zobrazoval slovní popis toho, co se se stromem děje. Operace nad stromy by měly mít možnost animace.

1.1.3 Testová část

V testové části dostane uživatel k dispozici náhodně vygenerovaný strom a operaci, kterou má provést. Následně bude veden ke správné odpovědi posloupností dotazů. Testová část by měla obsahovat možnost zobrazení správné odpovědi a generování nového úkolu. Stromy by měly být generovány náhodně, aby se uživatel nemohl naučit postup na konkrétních případech a místo toho si vžil vzor postupu na jakémkoliv stromu daného druhu.

1.2 Podobné aplikace

Hodnocení aplikací s podobným zadáním je obtížný úkol. Žádná z vyzkoušených aplikací neobsahovala všechny tři části ze zadání práce. Nejčastější jsou pak práce, které se věnují vizualizaci jednoho stromu.Situaci ještě více komplikuje skutečnost, že mnoho vizualizací bylo naprogramováno jako internetový *Java* applet, které jsou v dnešní době prohlížeči hojně blokovány z bezpečnostních důvodů. Tato řešení jsou většinou kvalitně provedené s graficky úhlednými animacemi. Jen málokdy však obsahují i příslušnou teorii či možnost výběru jiného druhu stromu v rámci jedné aplikace. Aplikace, která by umožňovala jakýmkoliv způsobem testovat uživatelovi vědomosti o stromech nebyla nalezena. Ukažme si proto alespoň pár aplikaci, které se věnují zobrazení a animování stromů.

1.2.1 AVL Tree visualization, Arsen Gogeshvili

Jedná se o internetovou aplikaci z roku 2007 postavenou na platformě *Adobe Flash* zobrazující AVL strom.
Dostupné z http://qmatica.com/DataStructures/Trees/AVL/AVLTree.html.

Klady

- \+ Kvalitní plynulá animace.
- \+ Zobrazuje text s provedenými kroky.
- \+ Obsahuje nápovědu.
- \+ Rozsáhlé nastavení zobrazení stromu a animace.

Zápory

- – Chybí možnost testování znalostí.
- – Teorie je zastoupena pouze odkazem na vhodnou literaturu.

Hodnocení

Velice povedená aplikace, které se toho nedá moc vytknout. Zamrzí absence teoretické části a jiných stromů.

1.2.2 RBTree Animation, Y. Daniel Liang

Velice jednoduchá internetová aplikace zobrazující červeno-černý strom. Autor má na svých stránkách i další druhy stromů.

Dostupné z http://cs.armstrong.edu/liang/animation/web/RBTree.html.

Klady

- + Více druhů stromů dostupných ze stránek autora.

Zápory

- – Malá škála operací.
- – Žádné animace.
- – Chybí slovní popis provedených kroků.
- – Oznamování pomocí dialogů.

Hodnocení

Spíše horší řešení dané problematiky.

1.2.3 Software pro podporu výuky datových struktur, Vladimír Dosoudil

Závěrečná práce na téma výuky datových struktur. Autor demonstruje algoritmy na Splay stromu (samovyvažovací BST). Jedná se o desktopovou aplikaci.

Dostupné z http://is.muni.cz/th/60914/fi. _b/.

Klady

- + Obsahuje pět DEMO příkladů.
- + Možnost krokování a pozastavení animace.
- + Možnost měnit horizontální a vertikální posun stromu.
- + Obsahuje 3 jazykové mutace (čeština, angličtina, slovenština).
- + Pseudokódy operací s komentářem.

Zápory

- − Při zmenšení okna podivné překrývání prvků.
- − Absence okna s výpisem kroků.
- − Neintuitivní uživatelské rozhraní
- − Absence možnosti testování znalostí.

Hodnocení

Práce má podobné zadání, jako jedna z mála aplikací obsahuje pseudokód s komentářem. Možnost pozastavení animace je dalším pozitivem.

1.2.4 Závěr

Existují práce, které plní alespoň část zadání. Některé jsou implementované lépe, některé hůře. Nejvíce jsou zastoupeny aplikace věnující se druhé části práce, teoretická část nalezneme jen některých případů, možnost prověřit uživatele testem se pak neobjevuje. Neexistence aplikace, která by spojovala všechny tři části (teoretickou ,praktickou část a testování), je dostatečnou motivací pro vznik této práce.

1.3 Diagram případu užití

Na základě zadání práce a specifikace požadavků vznikl zjednodušený diagram případu užití1. Ten zachycuje interakci uživatele s aplikací.

Obrázek 1: Graf případu užití.

2 Programátorská dokumentace

V této kapitole se seznámíme s technologiemi použitými při vytváření aplikace. Dále si ukážeme způsob implementace aplikace, přehled balíků, tříd, nejdůležitějších metod a jejich stručný popis.

2.1 Použité technologie

V následujících podkapitolách jsou uvedeny informace o použitých technologiích a knihovnách, které využívá aplikace. Jako programovací jazyk byl vybrán jazyk *Java*. Grafické prostředí aplikace je vytvořeno pomocí knihovny *JavaFX*. Pro zobrazení teoretické části je použita komponenta *WebView* a webové knihovny a jazyky:

- MathML.
- Bootstrap.
- JavaScript.
- jQuery.
- Scrolling nav layout.
- Highlight plugin.

2.1.1 Java

Java patří do rodiny objektově orientovaných jazyků. První verze byla vydána v roce 1995, zatím poslední osmá verze v roce 2014. Podle *TIOBE Programming Community Index* [7] a *PYPL PopularitY of Programming Language* [6] se Java poslední měsíce objevuje mezi nejpopulárnějšími jazyky. Výhodou použití jazyka Java je jeho multiplatformnost ,rozsáhlá dokumentace, široká komunita (dostupnost velkého množství knihoven a nástrojů) a možnost spustit Java aplikaci bez instalace na stroji, který má nainstalovanou příslušnou verzi JVM [1]. Aplikace byla napsána ve verzi jazyka *8*. Pro psaní práce bylo použito vývojové prostředí *Netbeans* ve verzi *8.0*. *Netbeans* je open-source[2] multiplatformní vývojové prostředí pro jazyk *Java* i jiné jazyky (namátkou *PHP*, *javascript*, *C/C++*).

2.1.2 JavaFX

Grafické prostředí aplikace bylo vytvořeno za pomocí nové knihovny integrované do Javy 8 *JavaFX*. Tato knihovna je plnohodnotným nástupcem[8] knihovny *Swing*, která slouží k ovládání aplikace pomocí uživatelského prostředí. Z *JavaFX*

[1] Java Virtual Machine

[2] Česky otevřený software. Uživatelé mohou volně prohlížet, používat a upravovat zdrojový kód.

přináší několik nových prvků, které usnadňují tvorbu uživatelského prostředí aplikace:

- **FXML** - Značkovací jazyk založený na XML[3], ve kterém je možné vytvořit uživatelské prostředí separované od logiky aplikace.
- **CSS**[4] - Vzhled komponent prostředí (například tlačítka) je možné upravovat pomocí kaskádových stylů.
- **Vícedotykové ovládání** – Přidána podpora vícedotykového ovládání.
- **Standalone aplikace** – Možnost vytvoření balíku, který kromě samotné aplikace obsahuje i běhové prostředí Javy. Uživatel pak nemusí mít běhové prostředí nainstalováno a stejně může spustit aplikaci (nevýhodou je větší velikost).
- **WebView** – Webová komponenta umožnující zobrazení webových stránek v aplikaci.
- **Animace** – Vytváření animací je daleko jednodušší, než tomu bylo v knihovně *Swing*.

2.1.3 WebView

Webová komponenta uvedená v rámci knihovny *JavaFX*. Slouží k zobrazení HTML stránek. Je založen na open-source jádru *WebKit* WebKit [2]. Podporuje CSS, javascript a HTML5. V aplikaci slouží k zobrazení HTML stránek s výukovým textem. Podrobnější vysvětlení zvolení tohoto přístupu nalezneme v podkapitole Architektura aplikace, v části věnující se balíku Theory.

2.1.4 MathML

MathML je specifikace umožnující zapisování matematických výrazů a rovnic do webového dokumentu a jejich správné zobrazení uživateli. *MathML* spadá pod konsorcium *W3* [4]. Jedná se o značkovací jazyk, jenž matematické prvky obaluje do příslušných značek. Z dané rovnice pak vzniká *XML* kód, který je obtížně čitelný pro člověka, ale počítač je schopný ho převést na reprezentaci, která je pro uživatele přirozená. V aplikaci je použit *MathMl* pro zobrazení matematických definic a vzorců.

[3]Extensible Markup Language
[4]Cascading Style Sheets
[4]World Wide Web Consortium - vyvíjí webové standarty pro world wide web

2.1.5 Bootstrap

Soubor HTML,CSS,JS[5] nástrojů pro tvorbu responzivních, uživatelsky příjemných webových stránek. Pomocí tohoto nástroje je možné vytvořit stránky, které se budou přizpůsobovat velikosti okna a zařízení na kterém jsou prohlíženy. V rámci aplikace pro výuku stromů je pak výhodou použití již vytvořených komponent a stylů nástroje. Pokud by se někdy měla objevit aplikace na jiných než desktopových zařízeních, díky nástroji *Bootstrap* by bylo jednoduší přizpůsobovat učební text formátu daného zařízení. *Bootstrap* je volně dostupný pod MIT licencí[6] na stránce http://getbootstrap.com/.

2.1.6 Scrolling nav layout

Nástroj umožnující scrollování (rolování po stránce) na místo odkazu po kliknutí na odkaz. Nástroj je možné stáhnout ze stránky http://startbootstrap.com/template-overviews/scrolling-nav/ pod licencí Apache 2.0[7]. Samotná stránka *startbootstrap.com* pak obsahuje nástroje, rozvržení a témata pro webové stránky používající nástroj *Bootstrap*.

2.1.7 Highlight plugin

Webová komponenta WebView použitá v aplikaci nepodporuje funkci vyhledávání v textu. Tato funkcionalita je však podstatná a nedá se opomenout. Byl proto zvolen vhodný nástroj, který je implementuje. Jedná se o *jQuery* knihovnu *highlight-5* vytvořenou Johannem Burkardem. Volně k dispozici je na stránce autora http://johannburkard.de/blog/programming/javascript/highlight-javascript-text-higlighting-jquery-plugin.html pod MIT licencí.

2.2 Architektura aplikace

V této kapitole se podíváme na implementaci aplikace, její třídy a metody.

Jazyk *Java* nabízí pro oddělení prostoru jmen tzn. balíky(packages).[8] Celá aplikace pak bylo rozdělena na 4 hlavní balíky:

1. **Theory** - Obsahuje teoretickou část aplikace.
2. **Algorithms** - Obsahuje logickou část aplikace.
3. **Representation** - Obsahuje část aplikace nutnou pro zobrazení a animování stromů.
4. **Test** - Obsahuje část aplikace určenou pro testování znalostí uživatele.

[5] javascript
[6] svobodná licence vzniklá na Massachusettském technologickém institutu
[7] svobodné licence, jejímž autorem je Apache Software Foundation
[8] Uživatel si je může představit jako složky, ve kterých jsou uchovány třídy.

Na stejné úrovni s těmito balíky je pak ještě složka s obrázky, třída `EducTree` a CSS soubor `educTreeStyle.css`.

Třída EducTree

Je první třídou, která se inicializuje po spuštění aplikace. Dědí z abstraktní třídy `Application`. Po spuštění aplikace je zavolána metoda `start(final Stage stage)`, která inicializuje hlavní okno a udržuje ho, až do vypnutí aplikace. V metodě `start(final Stage stage)` se vytvoří stromový seznam, který je umístěn do levé strany okna a slouží jako hlavní navigace v aplikaci. Pravou část okna pak tvoří panel, do kterého je vkládán obsah na základě volání metod při výběru určité položky ze stromového seznamu. Pomocí metody `scene.getStylesheets().add(...);` je aplikaci přidělen kaskádový styl ze souboru `educTreeStyle.css`.

educTreeStyle.css

JavaFX umožňuje vývojářům přidávat kaskádové styly jednotlivým komponentám aplikace. Soubor `educTreeStyle.css` pak obsahuje tyto styly. Styl je možné definovat globálně pro všechny prvky daného typu nebo lokálně pouze pro prvky s příslušným identifikátorem (přiděluje se pomocí metody `prvek.setId("libovolne_jmeno")`. V CSS souboru pak k němu přistoupíme pomocí znaku mřížky # [9].

```
1  /*Změní barvu všech tlačítek v~aplikaci na černou, font na Arial a
       veliksot fontu na 12px. */
2  .button{
3     -fx-background-color: black;
4     -fx-font-family: "Arial";
5     -fx-font-size: 12px;
6  }
7
8   /*Změní barvu tlačítek s~identifikátorem btnClickMe na červeno. */
9  #btnClickMe{
10  -fx-background-color: red;
11 }
```

Zdrojový kód 1: Ukázka *JavaFX* kaskádového stylu.

2.2.1 Balík Theory

Původní učební text k teoretické části aplikace byl vysázen v typografickém prostředí LaTeX. Toto prostředí dovoluje sázet matematiku na vysoké úrovni a pomocí knihovny *TikZ* kreslit grafy a stromy. Při samotné implementaci textu

[9] Anglicky hash.

do aplikace bylo potřeba zvolit správné prostředky pro plnohodnotné zobrazení textu. Zdánlivě nejjednodušší řešení, a sice zobrazení textu ve formátu *pdf* [10], se ukázalo jako problematické. Samotná *Java* nepodporuje zobrazení *pdf*. Tento problém alespoň částečně řeší knihovny třetích stran [11]. Jejich použití je pak limitováno cenou, rychlostí a často i špatným zobrazením matematických značek. Další způsob zobrazení takto profilovaného textu je pak zobrazení formou obrázků. Takto zobrazený text se relativně rychle načítá a umožnuje zachování matematiky a obrázků, tak jak je vysází LaTeX. Na druhou stranu interakce uživatele s textem je značně limitována. Uživatel nemůže označovat text, kopírovat ho, ani v něm vyhledávat. To může vést k frustraci uživatele a nechuti používat danou aplikaci. Obrázky také musí být ve vysokém rozlišení, aby byl text dobře čitelný, což se podepisuje na velikosti aplikace. Jako nejlepší řešení se ukázalo zobrazení výukového textu za pomocí jazyka *HTML*[12]. Takové stránky lze zobrazit pomocí komponenty *JavaFX* `webEngine`. Text LaTeXové verze byl překopírován do jazyka HTML a opatřen příslušnými HTML značkami a kaskádovým stylem pro lepší orientaci uživatele. Obrázky grafů a algoritmy pak byly převedeny do formátu *png* a vloženy na stránky. Jediným problémem tak zůstala matematika, jejíž LaTeXová podoba není v jazyce HTML akceptovaná. Tento problém zprvu řešila knihovna *MathJax*, která je schopná na místě převést LaTeXovou matematiku do formátu přijatelného pro HTML stránky. Toto řešení však fungovalo pouze při připojení do internetové sítě. Konečné řešení tak je použití jazyka *MathML*[13], do kterého byly převedeny LaTeXové formule. Text byl vypracován s pomocí odborné literatury. Použité zdroje jsou uvedeny na stránkách s textem. Pseudokódy použité v teorii jsou založeny na algoritmech z knihy *Introduction to algorithms*[1]

Samotný balík `Theory` pak obsahuje třídu `Theory`, HTML stránky s jednotlivými kapitolami učebního textu, s nápovědou, stránkou o aplikaci, adresář s obrázky, adresář s kaskádovými styly a adresář s knihovnami používanými HTML stránkami. Dále je v tomto balíku přítomná stránka s nápovědou a stránka s informacemi o aplikaci.

Třída Theory

Třída `Theory` obsahuje konstruktor s parametrem, který je volán ze třídy `EducTree`. Parametr určuje, která HTML stránka bude načtena do webového prohlížeče .

2.2.2 Balík Algorithms

V balíku `Algorithms` najdeme celkem 8 tříd, 2 třídy pro každý použitý strom. Jedna třída reprezentuje uzel daného stromu, druhá pak samotný strom a operace nad ním vedené. Tyto třídy pak tvoří logickou páteř reprezentace stromů v aplikaci.

[10] Portable document format.
[11] Nejznámější JPedal https://www.idrsolutions.com/jpedal/.
[12] HyperText Markup Language
[13] Mathematical Markup Language

Každá třída uzlu obsahuje jak atributy popsané v teoretické části (jako jsou klíč, odkaz na rodiče uzlu, odkaz na potomky atd.), tak atributy nesoucí informace o poloze uzlu na ose x a y, barvu uzlu, počáteční a koncovou souřadnici přímky spojující uzel s jeho rodičem a jiné informace důležité pro správné zobrazení uzlu po grafické stránce.

Třídy stromů pak obsahují odkaz na kořen stromu, díky kterému je možné manipulovat s celým stromem a algoritmy reprezentující operace nad daným druhem stromu. Popis algoritmů použitých v aplikaci se pak nachází v teoretické části. Následuje seznam tříd s výpisem nejdůležitějších metod a atributů.

Třída BSTStructure

Reprezentuje uzel binárního vyhledávacího stromu. Kromě ukazatele na uzel rodiče, potomků a číslo klíče má atributy `line` a `stack`. Atribut `stack` obsahuje `StackPane`, kontejner, který obsahuje `Circle` a `Text` v určitém pořadí na ose z. Tím je tvořena grafická reprezentace uzlu. Skládá se z kruhu určité barvy, na němž je text obsahující klíč uzlu. Atribut `line` obsahuje přímku spojující uzel s jeho rodičem.

Třída AVLStructure

Třída `AVLStructure` reprezentuje uzel AVL stromu. Na rozdíl od třídy `BSTStructure` obsahuje navíc atribut `balance`, který uchovává balanční hodnotu daného uzlu.

Třída RBTStructure

Třída `RBTStructure` reprezentuje uzel červeno-černého stromu. Na rozdíl od `BSTStructure` obsahuje navíc atribut `color` výčtového typu `TypeOfColor` udávající, jestli je uzel černý nebo červený a atribut `dummyNode`, který nabývá hodnoty true nebo false v závislosti na tom, jestli je vnitřní uzel stromu nebo list, který má být zobrazen pro potřebnou animaci. Zajímavostí je pak konstruktor třídy, který je třikrát přetížen pro různé situace.

Třída BStructure

Třída `BStructure` reprezentuje uzel B-stromu. Obsahuje více atributů, než třídy uzlů binárních stromů. Uveďme ty nejdůležitější:

keys - Seznam obsahující klíče daného uzlu.

childs - Seznam obsahující ukazatele na potomky daného uzlu.

parent - Ukazatel na rodiče uzlu.

group - Kontejner uchovávající kontejnery jednotlivých prvků v uzlu.

leaf - Pravdivostní hodnota, true pokud je uzel list, jinak false.

r - Maximální počet prvků v uzlu.

p - Minimální počet prvků v uzlu.

Třída BSTree

Třída `BSTree` představuje BST. Jejím jediným atributem je `root` ukazující na uzel kořene stromu. Důležité metody jsou:

- `insert(int key)`

Vložení nového uzlu s klíčem `key`.

- `find(int key)`

Nalezení uzlu s klíčem `key`.

Třída AVLTree

Třída `AVLTree` představuje AVL strom. Důležité atributy:

root - Ukazatel na uzel kořenu.

listOfString - Seznam řetězců. Jsou do něj ukládány slovní popisy toho, co se se stromem děje při provádění operací bez animace. Následně je vyzvednut třídou reprezentující strom a záznamy jsou ukázány uživateli.

Zajímavé metody jsou:

- `updateBalance`

Slouží k opravení vyváženosti AVL stromu po přidání uzlu do stromu. Metoda pak volá metody `updateBalanceLeft` a `updateBalanceRight` v závislosti na tom, jestli byl uzel přidán do levého nebo pravého podstromu.

- `updateBalanceR`

Slouží k opravení vyváženosti AVL stromu po odebrání uzlu, volá metody `updateBalanceLeftR` a `updateBalanceRightR` v závislosti na tom, jestli byl uzel odebrán z levého nebo pravého podstromu.

- `leftRotation`

Provede levou rotaci kolem uzlu `x`. V návratové hodnotě vrací ukazatel na nový kořen podstromu, kolem kterého byla prováděna rotace.

- `rightRotation`

Obdobně jako levá rotace, akorát pro pravou stranu.

- `insertWithoutRebalancing`

Vloží do stromu uzel, ale neprovede vyvážení stromu. Je volána při vytváření přesné kopie stromu nebo v různých cvičeních.

- `LinkedList<Integer> deleteWithoutRebalancing(AVLStructure node))`

Odstranění uzlu ze stromu bez vyvážení. Volá se ze cvičení na odebrání uzlu z AVL stromu. Jako návratovou hodnotu má seznam obsahující informace o tom, který uzel byl vymazán, a který uzel byl jeho rodič.

Třída RBTree

Třída `RBTree` představuje červeno-černý strom. Reprezentace červeno-černého stromu je speciální zavedením speciálního uzlu `nil` zastupujícího hodnotu `NULL`. Tento uzel je zároveň zástupnou hodnotou pro list. Důležité atributy:

root - Ukazatel na uzel kořenu.

listOfString - Seznam řetězců, jsou do něj ukládány slovní popisy toho co se se stromem děje při provádění operací bez animace. Následně je vyzvednut třídou reprezentující strom a záznamy jsou ukázány uživateli.

nil - ukazatel na speciální černý prázdný uzel, tento uzel supluje hodnotu `NULL`.

Zajímavé metody jsou:

- `RBTree()`

Konstruktor stromu přiřazuje do atributu `nil` ukazatel na černý uzel s hodnotami `NULL` na místě ukazatele na rodiče a potomky.

- `insertRebalance`

Volána na konci metody přidání uzlu. Stará se o vyvážení stromu po přidání uzlu podle algoritmu uvedeném v teoretické části. Vyvážení je docíleno vhodnými rotacemi a přebarvováním uzlů.

- `deleteRebalance`

Volána na konci metody odebrání uzlu, pokud byl odstraněný uzel černý. Stará se o vyvážení podle algoritmu uvedeném v teoretické části. Vyvážení může spadnout do 4 různých případů pro levou stranu a 4 pro pravou.

Třída BTree

Třída `BTree` představuje B-strom. Důležité atributy:

root - Ukazatel na uzel kořenu.

treeMaxDegree - Hodnota řádu stromu. Do atributu je přidělena při vytvoření instance stromu na základě přijímaného parametru.

Zajímavé metody jsou:

- `public Pair<BStructure, Integer> find(BStructure node, int key)`

Metoda pro hledání prvku. Jako parametry přijímá ukazatel na uzel, ve kterém se má začít hledání a klíč hledaného prvku. Návratovou hodnotu pak tvoří instance vnořené třídy `Pair<T,V>`, do které na místo prvního argumentu ukládáme ukazatel na uzel ve kterém je nalezený prvek. Jako druhý argument uvádíme index, na kterém je daný prvek uložen. Pokud hledaný podstrom prvek neukazuje, pak je na místo indexu uložena hodnota `NULL`.

- `insert`

Metoda pro vložení prvku do stromu. Volá výše uvedenou metodu `find` pro nalezení uzlu, do kterého prvek patří. Pokud je po přidání prvku uzel větší než je maximální povolená hodnota, pak je volána metoda `split(BStructure currentNode)`, která rozdělí uzel a přesune jeho prostřední prvek do rodiče.

- `remove`

Metoda pro odstranění prvku ze stromu. Pokud je odstraňovaný prvek v listu stromu, pak volá metodu `removeKeyFromLeaf(BStructure node,int key)` s ukazatelem na uzel a hodnotou prvku. Pokud odstraňovaný prvek není v listu, pak je pomocí metody `findMinimum(BStructure node)` volané s pravým podstromem, nalezen nejmenší prvek z pravého podstromu uzlu a ten je následně zkopírován na místo odstraňovaného prvku a vymazán z listu pomocí metody `removeKeyFromLeaf`. Pokud po odstranění uzlu klesne počet prvků pod minimum je volána metoda `rebalance`.

- `rebalance`

Metoda pro opravení B-stromu, pokud má jeden z jeho uzlů nedostatečný počet prvků. Pokud má takový uzel přímého sourozence s nadbytkem uzlů volá se metoda `leftTransfer` pokud je sourozenec levý nebo nebo `rightTransfer` pokud je sourozenec pravý. Pokud má přímý sourozenec přesně minimum prvků, pak se volá metoda `mergeLeft` nebo `mergeRight`. Levá nebo pravá metoda se volá podle toho, jestli je daný přímý sourozenec od uzlu vlevo nebo vpravo.

- `copyTree`

Metoda sloužící ke kopírování stromu do nově vytvořeného stromu. Postupně vkládá uzly ze stromu z parametru `bTree` do stromu z parametru `copyOfTree`. Volá se při ukládání stromu do historie undo nebo pokaždé, když je potřeba uchovat současnou podobu stromu.

2.2.3 Balík Representation

Balík obsahující balíky jednotlivých stromů. V každém nalezneme třídu nezbytnou pro grafické znázornění daného stromu a třídy řídící jednotlivé operace nad stromy. Speciální balík je pak `binaryTreeManager` obsahující rozhraní pro operace hledání a průchodů u binárních stromů (BST, AVL strom, červeno-černý strom). Základ každé reprezentace stromu je třída `TREErepresentation` [14]. Konstruktor této třídy vytváří ve spodní části přiděleného okna ovládací panel, který obsahuje prázdný box, ve kterém se zobrazuje slovní popis právě prováděné operace, textové pole pro zadání hodnoty klíče, tlačítka s operacemi, které jdou nad jednotlivým stromem provádět a konečně ovládání týkající se animací. Do horní části okna se pak vykresluje strom. Obsluha tlačítek s jednotlivými operacemi nad stromy pak volá příslušné metody tříd, které dohlíží nad danou operaci. Například obsluha tlačítka `btnAddNode` při zmáčknutí volá metodu `addNode` třídy `addNodeManager`.

Animace jednotlivých operací jsou řešeny pomocí drobných animací, které jsou spuštěny jedna za druhou. Například při hledání uzlu vytváříme animaci pro pohyb `TranslateTransition` pro každou cestu od jednoho uzlu ke druhému. Tyto drobné animace jsou vloženy do sekvenční animace `SequentialTransition`. Tato animace po spuštění přehrává postupně všechny animace do ní vložené v pořadí, v jakém byly vloženy. Díky tomuto přístupu můžeme kdykoliv pomocí metody `pause()` třídy `SequentialTransition()` zastavit animaci bez zjišťování, která drobná animace zrovna běží. V aplikaci jsou pak zastoupeny tyto druhy animací:

FadeTransition - V daném času zprůhledňuje/zneprůhledňuje daný objekt průběžnou aktualizací jeho atributu `opacity`.

FillTransition - V daném čase mění barvu daného objektu.

ParallelTransition - Umožňuje spuštění více animací současně.

ScaleTransition - V daném čase zvětšuje/zmenšuje objekt průběžnou aktualizací jeho `scaleX`, `scaleY`, `scaleZ` atributů.

SequentialTransition - Umožňuje spuštění více animací za sebou.

StrokeTransition - V daném čase mění barvu obrysu objektu.

Pro zpřehlednění dokumentace dále uvedeme popis `TREErepresentation` zvlášť pro binární stromy a zvlášť pro B-stromy. Následně budou zmíněny rozhraní pro vyhledávací a průchodové operace binárních stromů a třídy řídící vložení a odebrání pro všechny stromy.

[14] místo TREE dosaďme RBTree pro červeno-černé stromy, BTree pro B-stromy, AVLTree pro AVL stromy, BSTree pro BST.

Třídy TREErepresentation pro binární stromy

Jednotlivé implementace třídy `TREErepresentation` se strom od stromu liší podle specifikací a potřeb daného stromu. Obecné zásady metod jsou však stejné:

- `printTree(Structure node)`

Pro uzel na vstupu vytvoří `StackPane`, který reprezentuje daný uzel vizuálně. `StackPane` je poté uložen do atributu daného uzlu a přidán do vykreslovací plochy. Na závěr metoda volá `equalizeTree` a případně `moveTreeToRight` či `moveTreeToLeft`.

- `printAllTreePrep()`

Vytvoří novou instanci stromu, do které postupně vkládá skrz rekurzivní metodu `printAllTreeStructure node)` všechny uzly z původního stromu. Tak se tiskne celý strom, kde každý uzel dostane nové koordináty.

- `createCopyOfTree()`

Vytvoří kopii stromu. Ta se hodí například při práci s undo funkcionalitou.

- `addTreeToUndo(Tree tree)`

Přidá do *undo* zásobníku strom na vstupu.

- `doUndo()`

Z vrcholu undo zásobníku je získán strom a pomocí metody `printAllTree` je vytisknut do hlavní plochy.

- `deleteTree()`

Vymaže aktuální strom.

- `skipAnimation()`

Dá povel k přeskočení animace. Animace pak nedobíhá do konce, místo ní je rovnou tisknut výsledek dané operace. Lze zavolat pouze, pokud existuje běžící animace.

- `addListenersToStackOfNodes()`

Každému uzlu přiřadí listener, který reaguje na kliknutí na daný uzel. V takovém případě je daný uzel zvýrazněn a jeho hodnota klíče je vepsána v textovém poli.

- `equalizeTree(Structure node)`

Prochází daný strom a hledá uzly, které jsou nad sebou. Stará se o to, aby byl strom roztažen do šířky a nikdy nedošlo ke kolizi dvou uzlů.

- `moveTreeToRight(Structure node, double difference)`

Pokud by přidání nového uzlu (nebo rotace u AVL a červeno-černých stromů) způsobilo posunutí stromu mimo levou stranu kreslícího plánu, bude strom posunut doprava.

Třída BTreeRepresentation

Třída slouží k reprezentaci B-stromu. Na rozdíl od tříd pro binární stromy v dolní části obrazovky najdeme skupinu přepínačů pro zvolení příslušného řádu stromu. Při výběru řádu je vymazán současný strom, proto je v takovém případě zobrazen dialog dotazující se uživatele, zda si je jistý. Zajímavé metody:

- `printTree`

Na rozdíl u metod u binárních stromů tiskne `printTree` celý strom. Uzlu je zde přidělen kontejner `Group`, ve kterém jsou umístěny jednotlivé prvky v kontejnerech `StackPane`. Celý uzel je pak ohraničen obdélníkem. Správné symetrické umístění jednotlivých uzlů na ose x se počítá pomocí základny z listů dosažitelných z daného uzlu.

- `findChildsCoords`

Pro uzel na vstupu najde startovní a konečnou pozici listů dosažitelných z daného uzlu. Tyto pozice pak uloží do listu na pozice 0 a 1 a vrátí návratovou hodnotou.

- `leafsSize`

Metoda počítá celkovou šířku poslední úrovně stromu (obsahující listy). Pomocí této hodnoty je pak počítána počáteční souřadnice x pro umístění kořene stromu.

Rozhraní FindNode<T,V>

Slouží k nalezení uzlu ve stromu. Dané rozhraní implementují třídy:

- BSTFindNodeManager
- RBTFindNodeManager
- AVLFindNodeManager

Výpis některých metod implementovaných třídami:

- `findNode`

Voláno z obsluhy tlačítka pro hledání prvku. Dále volá `animateSearchNode` nebo `findNodeWithoutAnimation` v závislosti na tom, jestli jsou zapnuté animace.

- `findNodeWithoutAnimation`

Vyhledání prvku bez animace.

- `animateSearchNode(V searchKey))`

Vyhledání prvků s animací.

Rozhraní FindMinimumMaximum<T>

Slouží k nalezení nejmenšího nebo největšího prvku ve stromu. Dané rozhraní implementují třídy:

- BSTMinimumMaximumManager
- RBTMinimumMaximumManager
- AVLMinimumMaximumManager

Výpis některých metod implementovaných třídami:

- `findMaximalNode`

Voláno z obsluhy tlačítka pro hledání nejmenšího či největšího prvku. Dále volá metody `animateSearchMinimumMaximumNode(boolean isMinimum)` nebo `findMaximumWithoutAnimation()` v závislosti na tom, jestli jsou zapnuté animace.

- `animateSearchMinimumMaximumNode(boolean isMinimum))`

Volá metody pro zahájení animace pro vyhledání nejmenšího nebo největšího prvku.

Rozhraní Walk<T>

Slouží k jednotlivých průchodům stromu do hloubky. Dané rozhraní implementují třídy:

- BSTWalkManager
- RBTWalkManager
- AVLWalkManager

Výpis některých metod implementovaných třídami:

- `walkIno`

Voláno z obsluhy tlačítka pro průchod stromem inorder. Dále volá metody `walkInoAnimation` nebo `walkInoWithoutAnimation` v závislosti na tom, jestli jsou zapnuté animace.

- `walkInoAnimation`

Metoda pro animování průchodu stromem inorder.

- `walkInoWithoutAnimation`

Metoda pro průchod stromem inorder bez animace.

Rozhraní a třídy pak obsahují metody i pro průchod postorder a preorder.

Rozhraní WalkBFS<T>

Slouží k průchodu stromu do šířky. Dané rozhraní implementují třídy:

- BSTWalkBFSManager
- RBTWalkBFSManager
- AVLWalkBFSManager

Výpis některých metod implementovaných třídami:

- `ParallelTransition createScaleAnimation(final T node))`

Vytvoří animaci, při které každý procházený uzel na chvíli zvětší svou velikost a následně se vrátí do původního stavu.

- `walkBFSAnimation`

Metoda pro animování průchodu stromem do šířky.

- `walkBFSWithoutAnimation(T node)`

Metoda pro průchod stromem do šířky bez animace.

Třída BSTAddNodeManager

Třída `BSTAddNodeManager` obstarává operace přidání prvku do binárního vyhledávacího stromu. Obsahuje jak metodu pro přidání prvku bez animace tak s ní. Při animování se používá označení aktuálního uzlu kruhem s transparentní výplní a barevným okrajem. Zajímavé metody:

- `moveCircle`

Metoda vybírá další prvek na cestě k umístění přidávaného prvku. Obsahuje animaci měnící barvu obrysu označení aktuálního uzlu z fialové na červenou. Dále volá metodu `moveAnimation`.

- `moveAnimation`

Metoda obsahuje animaci pro pohyb označení uzlu k uzlu o jednu úroveň níže. Volá zpátky metodu `moveCircle`. Metody se navzájem volají tak dlouho, dokud se označení nedostane na místo, na které se má uložit nový prvek. `moveAnimation`.

Třída BSTRemoveNodeManager

Třída `BSTRemoveNodeManager` slouží k obsluze operace smazání uzlu z binárního vyhledávacího stromu. Jako ostatní obsluhy operací obsahuje metody k animovanému i neanimovanému smazání uzlu. Zajímavé metody:

- `deleteAnimationOneChildFinal`

Metoda volaná na závěr animace smazání prvku, který má pouze jednoho potomka. Animace smazání je tvořena pomocí metod `FadeTransition`, které jednotlivé objekty v daném čase zprůhledňují a tím vytváří dojem jejich mizení. Pokud má mazaný uzel rodiče, pak mizí stejnou animací i přímka spojující potomka s mazaným uzlem. Za pomocí `FadeTransition` lze pak vytvořit i animaci s opačným efektem, tedy postupným zviditelněným objektu díky změně jeho atributu průhlednosti. Toho se využívá při animaci přidání nové přímky spojující potomka mazaného uzlu a rodiče mazaného uzlu.

- `subTreeMinimumAnimation`

Metoda volaná při mazání uzlu, který má dva potomky. V tomto případě metoda vytváří animace, které znázorňují cestu vedoucí k nejmenšímu uzlu jeho pravého podstromu.

Třída AVLAddNodeManager

Třída `AVLAddNodeManager` obstarává operace přidání prvku do AVL stromu. Obsahuje metody pro animování přidání prvku a následné vyvážení stromu pomocí rotací. Zajímavé metody:

- `moveNodeAfterRotation`

Metoda volaná z `updateBalanceLeft` a `updateBalanceRight` po animaci každé rotaci. Jako parametry bere odkazy na dva stromy. Tyto stromy se liší umístěním uzlů. Zatímco v `originalNode` je uložený uzel s polohou před rotací, v `newPositionNode` je uzel s polohou po rotaci. Následně je vypočítána animace pohybu, která přemístí starý uzel na pozice nového uzlu. Nové pozice uzlů jsou vypočítány díky pomoci metod z třídy `AVLTreeRepresentation` `printAllTreeLogicPrep`.

- `updateBalanceLeft`

Metoda pro animaci změn při opravování vlastnosti AVL stromu po přidání prvku. Metoda je napsaná pro případ, kdy je uzel levým potomkem svého rodiče. K metodě existuje zrcadlová metoda `updateBalanceRight`, která obstarává animace pro případ pravého potomka. Samotná metoda je rozdělena na několik animačních částí. Jako první je do seznamu s animacemi zapsána animace pro zobrazení změny balančního faktoru. Pokud se uzel po změně balančního faktoru dostane do nevyrovnaného stavu, je volána metoda pro příslušnou rotaci a s ní spojené animace.

Třída AVLRemoveNodeManager

Třída `AVLRemoveNodeManager` obstarává operace odstranění prvku z AVL stromu. Obsahuje metody pro animování odstranění prvku a následné vyvážení stromu vhodnými rotacemi. Tyto metody jsou podobné metodám z třídy `AVLAddNodeManager`.

Třída RBTAddNodeManager

Třída `RBTAddNodeManager` obstarává operace přidání prvku do červeno-černého stromu. Obsahuje metody pro přidání prvku a následné vyvážení stromu. Vyvážení je animované pomocí animací pro změnu barvy u přebarvování uzlu a animací pro pohyb u rotace.

Třída RBTRemoveNodeManager

Třída `RBTRemoveNodeManager` obstarává operace odebrání prvku z červeno-černého stromu. Je komplikovanější než třída pro přidání prvku, protože je potřeba v některých případech animovat i uzel, který byl odebrán ze stromu. ten pak již nemá svou grafickou reprezentaci. Proto je nutné vytvořit pro něj zástupný uzel, nazvaný `dummyNode`, který ho zastupuje při animování.

Třída BTreeAddNodeManager

Třída `BTreeAddNodeManager` obstarává operace vložení prvku do B-stromu. Pro vyvážení uzlu při překročení maximální velikosti po přidání prvku se volá metoda `splitRoot` či `splitNonRoot` v závislosti na tom, zda byl přeplněný uzel kořen.

Třída BTreeRemoveNodeManager

Třída `BTreeAddNodeManager` obstarává operace odstranění prvku z B-stromu. Při nedostatku prvků v uzlu jsou volány metody `leftTransfer` pokud má levý přímý potomek více než minimum prvků nebo `rightTransfer` pokud má uzel pravého sourozence více než minimum prvků. Pokud má přímého sourozence s minimálním počtem prvků volají se metody pro spojení těchto prvků `mergeLeft` či `mergeRight`. U tříd `BTreeAddNodeManager` a `BTreeAddNodeManager` se nejčastěji používá animace pro změnu obrysu uzlu, či prvku v něm uloženém.

2.2.4 Balík Test

Obsahuje balíky jednotlivých stromů. V každém balíku pak se nacházejí třídy s různými druhy testů. Při vytváření testů byl kladen důraz na interakci uživatele s aplikací, vedení uživatele ke správnému výsledku a variabilitu testů. Každý test v balíku je pak v rámci svého druhu náhodně generován. Různé testy pak mají různou povahu. U některých musí uživatel vypisovat správné odpovědi do

připravených textových polí, u jiných klikat na příslušné uzly stromu nebo vybírat odpověď na otázku z nabízených možností. Každý test pak obsahuje možnost zobrazení řešení a vygenerování nového problému.

Všechny stromy mají dvě třídy testů na přidání a odebrání prvku. BST navíc obsahuje testy na průchod stromem do hloubky i do šířky. AVL strom obsahuje navíc testy na ohodnocování balančních faktorů stromu a jednoduchý test na poznávání rotací. Následují jednotlivé třídy balíku s vybranými metodami.

Třída InorderExercise

Třída `InorderExercise` spravuje test na inorder průchod binárním vyhledávacím stromem. Kromě třídy `InorderExercise` obsahuje balík další 3 třídy věnující se průchodům stromem (preorder, postorder, do šířky). Tyto třídy jsou řešeny stejným způsobem jako třída `InorderExercise`, proto je nebudeme uvádět. V tomto testu je vytvořen strom o deseti uzlech. Uzly v tomto stromy jsou losovány náhodným generátorem s rozsahem od nuly do devatenácti. Uživatel má za úkol do deseti připravených textových polí vyplnit posloupnost, v jaké by prošel daný strom průchodem inorder. Při odeslání odpovědi pak správně vyplněná pole zezelenají, zatímco špatně vyplněná pole zčervenají. Správné řešení testu je uloženo v atributu `listOfInorder`.

Třída InsertExerciseBST

Třída `InsertExerciseBST` spravuje test na přidávání prvku do binárního vyhledávacího stromu. Uživatel má za úkol přidat uzel do náhodně generovaného stromu. Na `StackPane` uzlů jsou navázány události kliknutí myši. Uživatel má za úkol kliknout na uzel, který bude rodičem přidávaného prvku. Pokud uživatel vybere správný uzel, musí vybrat, jestli se prvek stane levým nebo pravým potomkem uzlu.

Třída RemoveExerciseBST

Třída `RemoveExerciseBST` spravuje test na odebírání prvku z binárního vyhledávacího stromu. Test je realizován sérii klikání na uzly. O současném úkolu informuje uživatele text v dolní části okna.

Třída BalanceFactorExercise

Třída `BalanceFactorExercise` spravuje test na poznávání balančních faktorů u AVL stromů. Balanční faktory u náhodně generovaného stromu jsou skryty otazníky. Uživatel k jednotlivým klíčům píše do textového pole čísla balančních faktorů. Při správné odpovědi je otazník nahrazen hodnotou.

Třída InsertExerciseAVL

Třída `InsertExerciseAVL` spravuje test na vyvážení AVL stromu po přidání prvku. Je vygenerován náhodný strom a do něj přidám náhodný prvek. Při tomto přidávání se využívá metoda umožnující přidat prvek bez vyvážení stromu. Uživatel je veden otázkami a odpovídá pomocí tlačítek z předem daných odpovědí, dokud není strom vyvážen.

Třída RemoveExerciseAVL

Třída `RemoveExerciseAVL` spravuje test na vyvážení AVL stromu po odebrání prvku. Je vygenerován náhodný strom a z něj odebrán náhodný prvek. Uživatel je veden otázkami a odpovídá pomocí tlačítek z předem daných odpovědí, dokud není strom vyvážen.

Třída RotationExercise

Třída `RotationExercise` spravuje test na rotace AVL stromu. Je vygenerována náhodná část nevyváženého stromu o třech uzlech se zadanými balančními faktory. Uživatel pak vybírá vhodnou rotaci.

Třída InsertExerciseRBT

Třída `InsertExerciseRBT` funguje obdobně jako třída pro testování AVL vyvážení po přidání prvku. Zde se jedná o vyvážení po přidání prvku do červeno-černého stromu.

Třída RemoveExerciseRBT

Třída `RemoveExerciseRBT` funguje obdobně jako třída pro testování AVL vyvážení po odebrání prvku. Zde se jedná o vyvážení po odebrání prvku z červeno-černého stromu.

Třída InsertExerciseBT

Třída `InsertExerciseBT` spravuje přidání prvku do B-stromu řádu 3. Generovaný strom obsahuje 20 prvků v rozmezí hodnot jedna až třicet. Uživatel odpovídá na otázky pomocí tlačítek a textových polí.

Třída RemoveExerciseBT

Třída `RemoveExerciseBT` spravuje odebrání prvku z B-stromu řádu 5. Generovaný strom obsahuje 20 prvků v rozmezí hodnot jedna až třicet. Uživatel odpovídá na otázky pomocí tlačítek a textových polí.

3 Uživatelská dokumentace

Uživatelská dokumentace obsahuje návod ke spuštění aplikace a k její obsluze. Popisuje jednotlivé ovládací komponenty a práci s nimi.

3.1 Instalace a spuštění

Před samotným spuštěním aplikace je potřeba stáhnout a nainstalovat *Java Runtime Environment*, běhové prostředí jazyka *Java* a to alespoň ve verzi *8.51*. Toto prostředí je možné stáhnout ze stránek výrobce ve verzi pro Windows, Linux i Mac OS[2] . Aplikaci samotnou není nutné instalovat. Stačí pouze zkopírovat z adresáře `/bin` soubor EducTree.java a spustit ho dvojklikem nebo příkazem `java -jar EducTree.java`. Uživatelé Windows mohou zkopírovat z adresáře `/bin` adresář `standalone` obsahující verzi aplikace se zabaleným běhovým prostředím. Při spuštění této aplikace uživatel nemusí mít nainstalované *Java Runtime Environment*.

Aplikace byla testována na platformě *Windows 8*, *Windows 8.1*, *Windows 10* a linuxové distribuci *Ubuntu*.

3.2 Hlavní okno

Při spuštění aplikace se uživateli zobrazí okno 2, v jehož levé části se nachází hlavní menu aplikace sloužící k navigaci, a v pravé části plocha aplikace, na které se zobrazují vybrané požadavky uživatele. Jako výchozí stránka je zde nastavena část teoretické části zabývající se grafy.

3.3 Hlavní menu

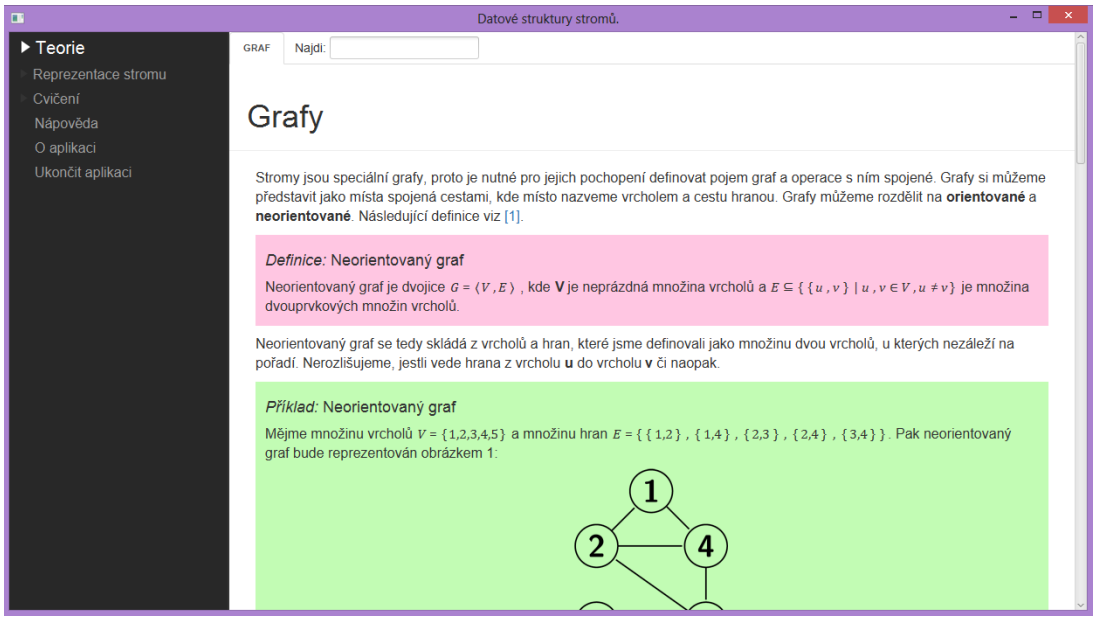
Hlavním navigačním prvkem celé aplikace je menu na levé straně, ve kterém můžeme vybírat z teorie reprezentace stromů nebo praktických cvičení, viz 3.

Dvojklikem na jednotlivé záložky je rozložíme a otevře se další specifičtější nabídka. V rámci teorie se se jedná o základy teorie grafů, stromů, binárních vyhledávacích stromů, AVL stromů, červeno-černých stromů a B-stromů. Pod reprezentací stromu můžeme vybírat ze stejných stromů, které následně můžeme vytvářet, mazat a provádět na nich základní operace. Poslední záložkou je pak záložka Cvičení obsahující možnost testování uživatele na různých testech pro všechny druhy popisovaných stromů. Dále se v menu nachází nápověda, stránka o aplikaci a možnost ukončení aplikace.

3.4 Horní menu teorie

Při rozkliknutí BST stromů, AVL stromů, červeno-černých stromů nebo B-stromů pod záložkou Teorie objeví se v hlavním okně učební text s charakteristikou

[2] http://www.oracle.com/technetwork/java/javase/downloads/index.html



Obrázek 2: Hlavní okno aplikace.

daného stromu. Pro další navigaci mezi jednotlivými kapitolami o daném stromu pak slouží horní vedlejší menu 4.

3.5 Operační menu binárních stromů

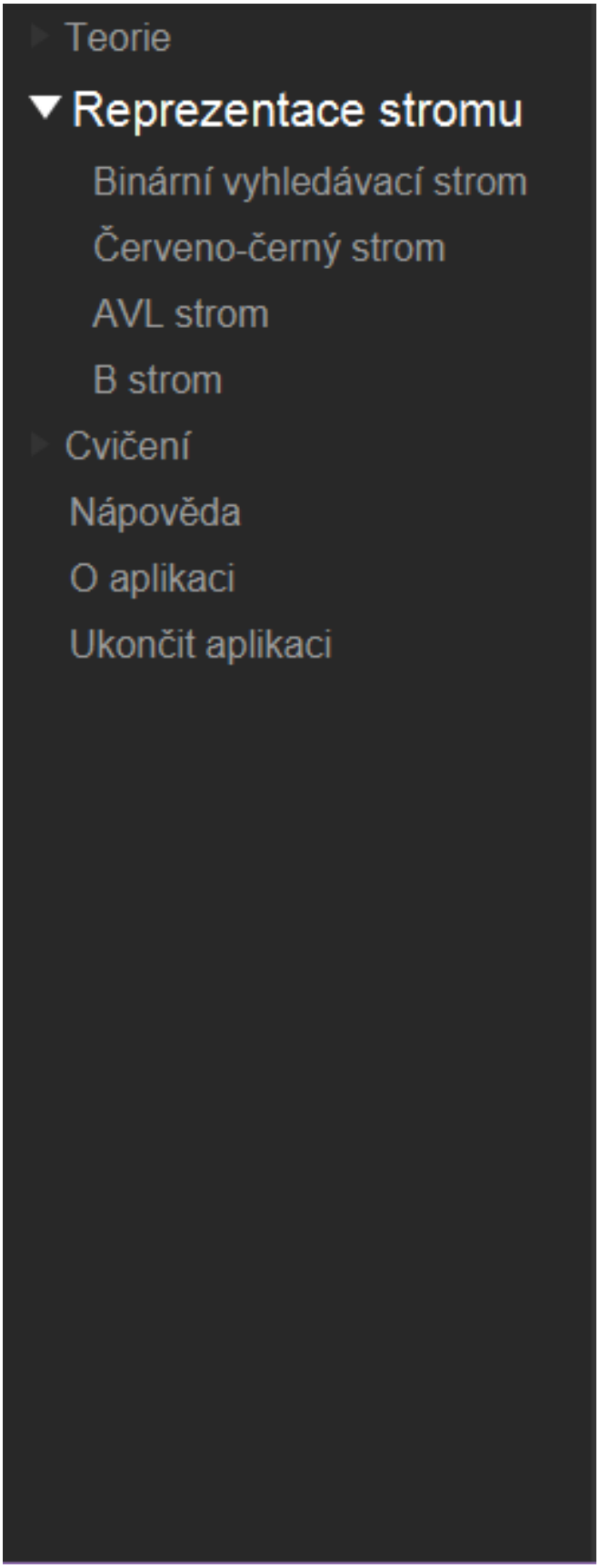
Při rozkliknutí jednoho z binárních stromů pod záložkou Reprezentace stromu se objeví ve spodní části obrazovky menu umožnující operace nad daným stromem 5. Dále obsahuje více kontrolních prvků, proto si je popíšeme tak, jak se nachází v menu zleva doprava.

Okno s výpisem

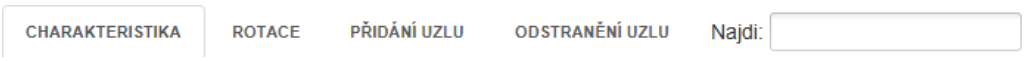
V levé části spodního menu se nachází okno, do kterého jsou vypisovány v reálném čase proměny daného uzlu. Toto okno má čistě informační hodnotu a uživatel s ním nemůže interagovat.

Tlačítka základních operací

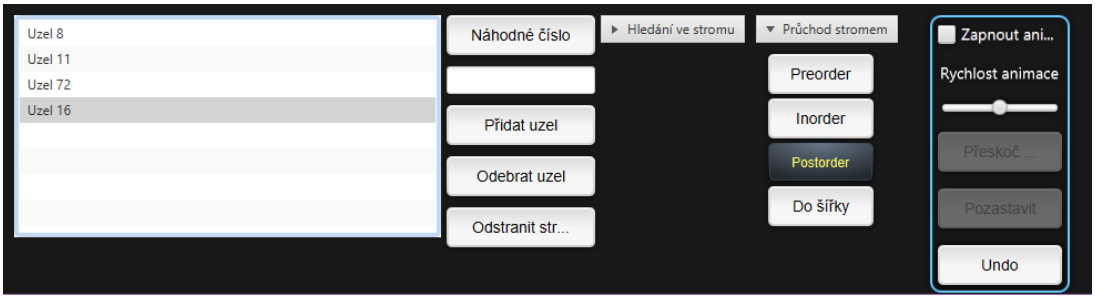
Tlačítka operací slouží k provádění samotných operací nad stromem. Pokud chce uživatel vložit nový prvek do stromu, musí nejprve do textového pole uvést jeho hodnotu nebo může zmáčknout tlačítko *Náhodné číslo*, které do pole dosadí náhodné číslo v rozmezí 0 až 99. Poté může uživatel zmáčknout tlačítko *Přidat uzel*, které zavolá obsluhu dané operace. Další možností je odebrání uzlu v z daného stromu. K tomu slouží tlačítko *Odebrat uzel*. Při odebrání uzlu je



Obrázek 3: Hlavní menu aplikace.



Obrázek 4: Horní vedlejší menu.



Obrázek 5: Operační menu binárních stromů.

nutné specifikovat, který uzel má být vymazán. To může být provedeno dvěma způsoby:

1. Vypsáním dané hodnoty do textového pole.
2. Kliknutím na daný uzel v reprezentaci stromu v horní části obrazovky.

Dale uživatel může odstranit celý strom pomocí tlačítka *Odstranit strom*.

Rolovací menu hledání ve stromu

Po rozkliknutí rolovacího menu *Hledání ve stromu* se objeví další tři tlačítka:

- Najít uzel
- Minimum
- Maximum

Tyto tlačítka slouží k vyhledávání ve stromu. Po zmáčknutí tlačítka *Minimum* nebo *Maximum* je nalezen nejmenší (největší) uzel stromu. Tlačítko *Najít uzel* hledá uzel, jehož hodnota je vypsána do textového pole nebo uzel, na který uživatel klikne.

Rolovací menu průchod stromem

Po rozkliknutí rolovacího menu *Průchod stromem* se objeví další čtyři tlačítka:

- Preorder
- Inorder

- PostOrder
- Do šířky

Každé z těchto tlačítek pak volá obsluhu, která provede daný průchod stromem podle zvoleného tlačítka.

Tlačítka animace

Úplně vpravo dolního menu se pak nachází kontrolní prvky ohraničené tyrkysovým obdélníkem. Tyto prvky pracují s konceptem animace a možností vrátit strom do předešlého stavu. Konkrétní prvky této části jsou:

Zapnout animace Box, který nabývá dvou hodnot. Může být zaškrtnutý nebo ne. Pokud ho uživatel zaškrtne, budou všechny operace nad stromem prováděny s animací.

Rychlost animací Posuvný jezdec, kterým jde regulovat rychlost animací. Při nastavení jezdce úplně vlevo budou animace nejpomalejší. Při nastavení jezdce úplně vpravo budou animace nejrychlejší. Ve výchozím nastavení je jezdec uprostřed. Rychlost animace se nedá měnit v průběhu animace.

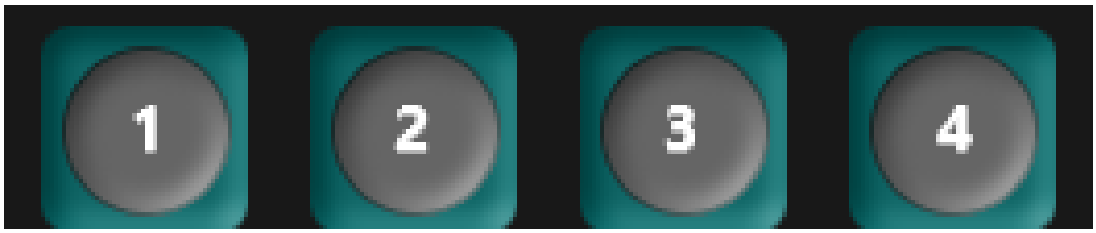
Přeskočit animaci Tlačítko s funkcí okamžitého ukončení animace. Při použití ukončí danou animaci v průběhu a rovnou zobrazí strom po skončení operace. Tlačítko nelze spustit, pokud žádná animace neběží.

Pozastavit/Spustit Tlačítko mění svůj název v závislosti na tom, v jakém stavu se nachází. Výchozí hodnota je slovo *Pozastavit*. V takovém stavu při zmáčknutí tlačítka dojde k pozastavení animace a změně názvu na *Spustit*. Při zmáčknutí tlačítka během druhého stavu, dojde k opětovnému spuštění animace a změně názvu na *Pozastavit*. Tlačítko nelze použít, pokud neběží žádná animace.

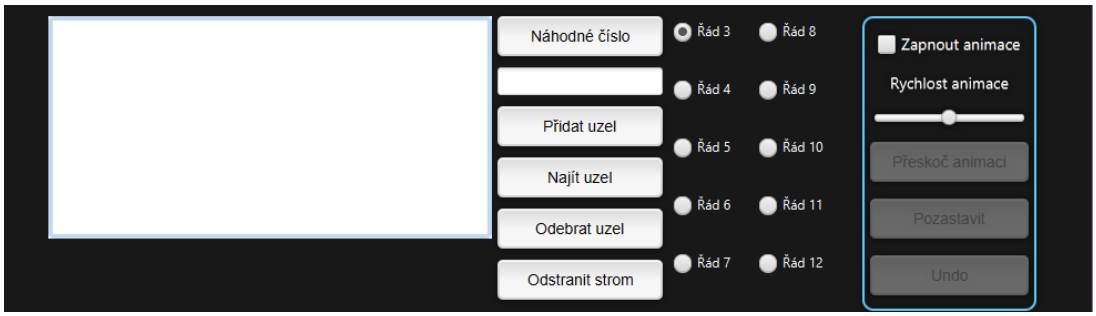
Undo Poslední tlačítko celého menu. Při zmáčknutí se vrátí strom do předešlého stavu před operaci, která ho nějákým způsobem pozměnila. Lze použít vícekrát, pokud byl daný strom vícekrát měněn.

Stav případů červeno-černých stromů

Červeno-černý strom, kromě výše zmíněných kontrolních prvků ovládacího menu, obsahuje sérii 4 kulatých indikátorů stavů nad oknem s výpisem6. Tyto stavy symbolizují vstupy do jednotlivých 4 případů vyvážení při přidávání či odstraňování uzlu. Pokud je zaplá animace, pak při vstupu do případu proběhne krátká animace rozsvícení daného indikátoru stavu. To společně se slovním výpisem pomáhá uživateli v orientaci mezi případy u vyvážení červeno-černých stromů.



Obrázek 6: Stav indikátorů případů u červeno-černých stromů.



Obrázek 7: Nástrojové menu B-stromů.

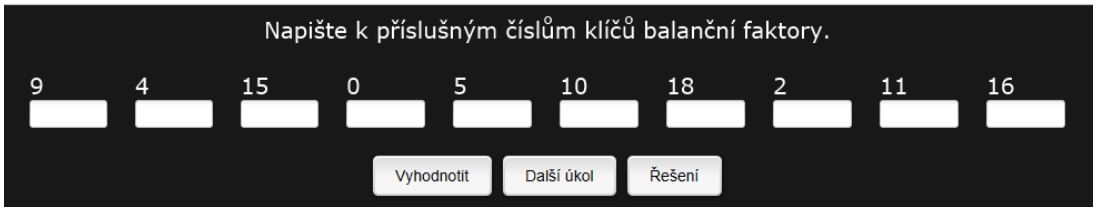
3.6 Nástrojové menu B-stromů

Operační menu u B-stromů je podobné operačnímu menu binárních stromů. Jediným rozdílem je nahrazení rolovacích menu pro hledání ve stromu a průchod stromem sérií dvanácti přepínačů7. Tyto přepínače udávají hodnotu řádu stromu. Zapnutý může být vždy jen jeden. Při změně přepínače je vymazán současný strom a vytvořen nový prázdný s daným řádem z přepínače. Protože takové chování je destruktivní a uživatel by si ho nemusel být vědom, je při změně přepínače v době, kdy má uživatel vytvořený strom s alespoň jedním uzlem, vyvolán dialog informující uživatele o smazání uzlu a čekající na jeho souhlas nebo zrušení přepnutí řádu.

3.7 Menu cvičení

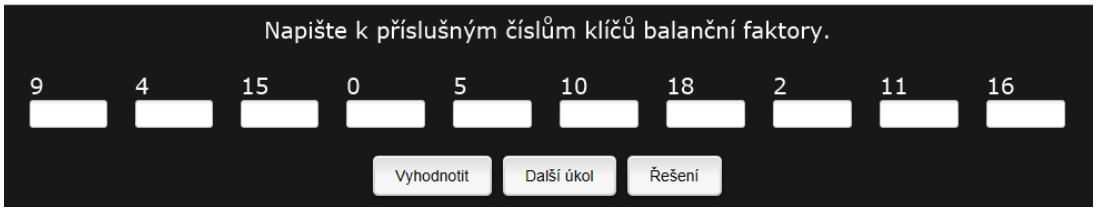
Jednotlivé testy pro testování uživatele nemají jednotné ovládací prvky. Uvedeme si proto pár příkladů, jak mohou tyto testy vypadat. V některých případech jako jsou například průchody stromem, je třeba do textových polí vypsat čísla uzlů ve správném pořadí podle instrukcí v horní části. Ve spodní části se nachází tlačítko pro vyhodnocení. Kolonky se zabarví do zelena/červena podle toho, jestli je odpověď správná/špatná. Tlačítko *Další úkol* vygeneruje další příklad a *Řešení* vepíše správné odpovědi 8.

Další druh cvičení probíhá výběrem správné odpovědi podle druhu otázky. Součástí jsou opět tlačítka pro nový úkol a správné řešení. Součástí některých

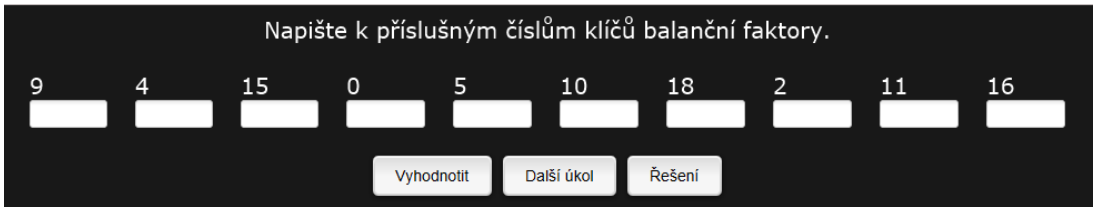


Obrázek 8: Cvičení průchodu binárním vyhledávacím stromem.

cvičení je i série dvou obrázku ukazující, zda je vaše odpověď správná či nikoliv 9. 10.



Obrázek 9: Cvičení na přidání uzlu do stromu.



Obrázek 10: Cvičení na odebrání prvku ze stromu.

Závěr

Výsledkem bakalářské práce je vzdělávací aplikace určená pro podporu výuky datových struktur stromů.Na základě specifikace požadavků byla aplikace rozdělena na tři části. První část obsahuje teoretické poznatky k výuce. Uživatel se zde seznámí s grafy, stromy a s reprezentacemi binárních vyhledávacích stromů, AVL stromů, červeno-černých stromů a B-stromů. Text obsahuje postupy algoritmů základních operací nad těmito stromy. V druhé části jsou implementovány použité algoritmy do podoby grafické reprezentace, ve které může uživatel tvořit výše zmíněné stromy a sledovat průběh operací nad nimi. Třetí část nabízí uživateli možnost otestovat své vědomosti v interaktivních testech nad náhodně generovanými stromy. Cílem této bakalářské práce bylo vytvoření vzdělávací aplikace pro podporu výuky datových struktur stromů. Aplikace skýtá příležitosti k možnému vylepšení v budoucnu. Jako první se nabízí rozšíření o další datové struktury stromů, například o 2-3-4 stromy nebo B+ stromy. Dalším možným vylepšením by bylo přeložení textu i aplikace do jiných jazyků. S tím souvisí i vytvoření způsobu pro upravování a přidávání učebních textů přímo v aplikaci. Uživatelské prostředí by mohlo být obohaceno o nastavení fontů písem a barevných schémat. Tato funkcionalita by vzhledem k možnostem JavyFX a CSS mohla být relativně snadno implementována.

A Obsah přiloženého CD/DVD

bin/
: Program EducTree, spustitelný přímo z CD/DVD. Adresář Standalone obsahující spustitelnou verzi pro Windows, která nepotřebuje ke svému běhu Java Runtime Environment.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové texty programu EducTree.

readme.txt
: Instrukce pro instalaci a spuštění programu EducTree, včetně všech požadavků pro jeho bezproblémový provoz.

U veškerých cizích převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD/DVD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.

Literatura

[1] Thomas H. Cormen .. [et]. Introduction to algorithms. 2nd. ed., 10th printing. Cambridge (Massachusetts): MIT Press, 2007. ISBN 0262531968.

[2] Open-source renderovací jádro WebKit LaTeX
Dostupné z: http://www.webkit.org/

[3] Typografický systém LaTeX
Dostupné z: http://www.latex-project.org

[4] Požadavky na státní závěrečnou zkoušku pro obor Informatika
Dostupné z: http://www.inf.upol.cz/downloads/studium/2015_INFv01_bc.pdf

[5] Požadavky na státní závěrečnou zkoušku pro obor Aplikovaná informatika
Dostupné z: http://www.inf.upol.cz/downloads/studium/2015_APLINFv01_bc.pdf

[6] PYPL PopularitY of Programming Language
Dostupné z: http://pypl.github.io/PYPL.html

[7] TIOBE Programming Community Index
Dostupné z: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[8] JavaFX Frequently Asked Questions
Dostupné z: http://www.oracle.com/technetwork/java/javafx/overview/faq-1446554.html