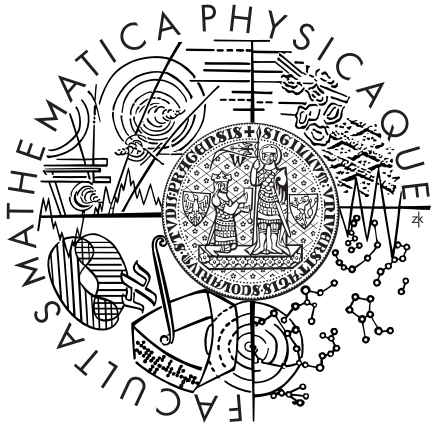
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Kamil Nezval

Chromatické stromy

Katedra softwarového inženýrství

Vedoucí diplomové práce: RNDr. Alena Koubková, CSc.

Studijní program: Informatika

Studijní obor: ISS softwarové systémy

Praha 2006

Děkuji svým rodičům za podporu při studiu a vedoucí mé diplomové práce RNDr. Aleně Koubkové za její vedení a cenné rady, které mi v průběhu práce poskytla.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 18.4.2006 Kamil Nezval

Obsah

Název práce: *Chromatické stromy*

Autor: *Kamil Nezval*

Katedra: *Katedra softwarového inženýrství*

Vedoucí diplomové práce: *RNDr. Alena Koubková, CSc.*

E-mail vedoucího: *Alena.Koubkova@mff.cuni.cz*

Abstrakt: *Červeno-černé stromy jsou obdobou binárních vyhledávacích stromů zaručující logaritmickou složitost svých operací i v nejhorším případě. Relaxovaná verze této datové struktury byla vyvinuta pro zrychlení operací a zvýšení míry souběžnosti v paralelním prostředí. Hlavní myšlenkou je oddělení vyvažování od aktualizace stromu. Cílem této diplomové práce je experimentálně porovnat standardní červeno-černé stromy a jejich jednotlivé relaxované varianty v neparalelním prostředí z hlediska chování na velkých datech.*

Klíčová slova: *binární vyhledávací strom, červeno-černý strom, relaxované vyvažování, oddělené vyvažování, chromatický strom, složitost*

Title: *Chromatic trees*

Author: *Kamil Nezval*

Department: *Department of software engineering*

Supervisor: *RNDr. Alena Koubková, CSc.*

Supervisor's e-mail address: *Alena.Koubkova@mff.cuni.cz*

Abstract: *Red-black trees are the binary search trees that guarantee logarithmic complexity also in the worst case. In order to speed up the response time of the operations and to allow a high degree of concurrency in parallel environment, relaxed balancing was introduced. The main idea is to uncouple the rebalancing from the updating. The aim of the diploma thesis is experimental comparison of standard red-black trees and three relaxed balanced versions in non-parallel environment while working with large data.*

Keywords: *binary search tree, red-black tree, relaxed balancing, uncoupled rebalancing, chromatic tree, complexity*

1 Úvod

Binární vyhledávací stromy jsou poměrně dobře známou a často používanou datovou strukturou. Časová složitost je úměrná jejich výšce, a proto bylo vyvinuto několik druhů úprav zaručující logaritmickou výšku stromu a tím i složitost operací $O(\log n)$. Mezi nejfrekventovanější patří například *AVL stromy*, *červeno-černé stromy* nebo *(a,b)-stromy*.

První návrh červeno-černých stromů, které jsou předmětem této práce, představil v roce 1972 Bayer pod názvem *symetrické binární B-stromy*. Označení červeno-černé stromy zavedli v roce 1978 ve své publikaci [3] Guibas a Sedgewick. Jejich myšlenka zaručuje zachování složitosti $O(\log n)$, kde n odpovídá počtu prvků ve stromě, a provedení maximálně tří jednoduchých rotací při aktualizaci stromu.

Za účelem zrychlení odezvy vykonávaných metod *insert*, *delete* a *search* a poskytnutí vyšší míry souběžnosti v paralelním prostředí, byla navržena představa relaxovaného vyvažování. Místo požadování okamžitého obnovení vyváženosti stromu po každé provedené aktualizaci, může být tato transformace odložena na pozdější dobu a lze tak obsloužit jiné žádosti. Vyvažovací funkce nejčastěji běží na pozadí všech procesů. Oddělené vyvažování od aktualizace stromu bylo poprvé zmíněno v [3] pro červeno-černé stromy. První skutečné řešení přinesl Kessels v roce 1984 na *AVL* stromech, ale umožňovalo pouze operaci *insert*. Rozšíření předchozí verze bylo později prezentováno v [14]. Relaxovaná verze červeno-černého stromu byla předložena v roce 1991 v [12]. Od té doby vzniklo několik studií popisujících relaxovaně vyvážené červeno-černé stromy, hojně označované jako *chromatické stromy*.

Jak jsme již naznačili, oddělení vyvažovací úlohy od aktualizace stromu bylo navrženo s ohledem na vylepšení chování této datové struktury v prostředí s více procesy. Cílem této diplomové práce je shrnout informace o jednotlivých verzích relaxovaných červeno-černých stromů a také experimentálně zhodnotit vhodnost takto upravených algoritmů v neparalelním prostředí. Zda je efektivní vyvažování odložit nebo provést po každé aktualizaci hod-

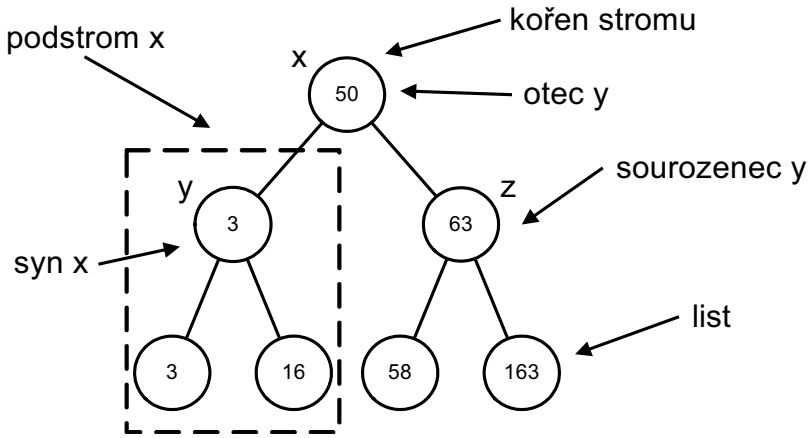
not ve stromu. A také zjistit výhodnost různých vyvažovacích period.

V kapitole 2 si uvedeme základní pojmy potřebné v dalším textu. Následující část popisuje binární vyhledávací stromy a v kapitole 4 se dostaneme k popisu vlastních červeno-černých stromů a relaxovaných verzí z nich vycházejících. Druhá polovina práce označená číslem kapitoly 5 je věnována experimentům a jejich výsledkům.

2 Základní pojmy a definice

V této kapitole připomeneme základní pojmy, s nimiž budeme pracovat v celém textu.

Strom je složen z uzlů spojených orientovanými hranami, které nesmí tvořit cykly. Obsahuje jeden odlišný uzel, který nazýváme *kořen*. Uvažujme uzel y ve stromu různý od kořene. Libovolný uzel x na cestě z kořene do y se nazývá *předchůdce* (předek) uzlu y. Jestliže x je předchůdce y, říkáme, že y je *potomek* x. Pokud poslední hrana na cestě z kořene do y je hrana (x, y), tak se uzel x nazývá *rodič* (otec) y a y je *dítě* (přímý potomek, syn) x. Dva uzly mající stejného rodiče označujeme jako *sourozence* (bratry). Uzel bez potomků se nazývá *list*, nelistový uzel je vnitřní uzel.



Obr. 2.1

Hloubka uzlu y je délka cesty z kořene stromu do y. Největší hloubka uzlu ze stromu se nazývá *výška* stromu.

Velikost stromu je počet listů ve stromě, značíme $|T|$.

Každý uzel obsahuje klíč, ukazatele na svého otce a na dva syny (v případě listu mají hodnotu NULL).

V této práci bereme v úvahu pouze stromy podle definice z následující kapitoly (listově orientované binární vyhledávací stromy).

3 Binární vyhledávací stromy

Základ červeno-černých stromů vychází z binárního vyhledávacího stromu (také označován zkratkou BVS), a proto si na začátek uvedeme jeho definici a dále se seznámíme s operacemi *insert*, *delete* a *member*.

Definice 1. Listově orientovaný binární vyhledávací strom je strom, pro který platí:

- každý uzel má dva nebo žádného následníka
- data jsou uložena pouze v listech – vnitřní uzly stromu obsahují pouze takzvané směrovače (angl. routers), které slouží pro určení správné cesty při průchodu stromem
- všechny uzly v levém podstromu vrcholu v obsahují menší nebo stejné hodnoty jako je hodnota v uzlu v a uzly v pravém podstromu mají větší hodnoty než uzel v.

Poznámka. Vnitřní uzly obsahují data stejného typu jako listy, nicméně tyto hodnoty nemusí existovat jako klíče v listech. Důvod je prostý. Nechceme totiž aktualizovat tyto interní uzly při každém odstranění listu ze stromu.

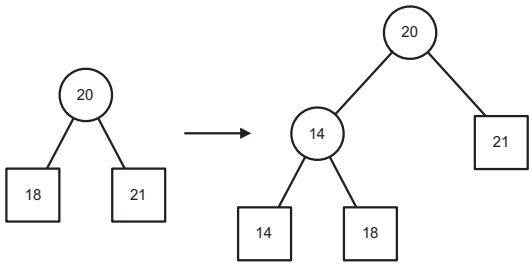
3.1 Operace MEMBER

Operace *member* (v některých literaturách označovaná jako *find* nebo *search*) slouží k vyhledání prvku ve stromu. Funkce postupně prochází strom od kořene k listům na základě porovnání hodnoty uložené v uzlu a hledaného klíče. Cestu stromem tedy určují vnitřní uzly. Podle definice binárního vyhledávacího stromu levý podstrom každého vnitřního uzlu obsahuje hodnoty menší nebo rovny hodnotě klíče v uzlu (viz definice 1). Pokud algoritmus dojde do listu stromu a zadaná hodnota se shoduje s klíčem v listu, je požadovaný uzel nalezen. V opačném případě není vyhledávaný klíč ve stromě obsažen a funkce končí neúspěchem.

Poznámka. Při použití této funkce v metodě *insert* (nebo *delete*) provedeme drobnou korekci. Návratovou hodnotou je vždy list, ve kterém skončil průchod stromem. Pokud se jeho hodnota liší od hledaného klíče, dostáváme pozici ve stromě, na jejíž místo připojíme nový list s požadovanou hodnotou. Více uvedeme v popisu operace *insert* a *delete* v následujících odstavcích.

3.2 Operace INSERT

Prvním krokem pro vložení nového klíče do BVS je nalezení jeho pozice mezi listy. Za tímto účelem použijeme upravenou verzi operace *member*, která je popsána v poznámce na konci předchozího odstavce. Pokud je hodnota ve stromu nalezena, operace končí. Jinak nalezený list nahradíme novým vnitřním uzlem (nazveme jej v) se dvěma listy. Hodnota levého syna uzlu v je nastavena na menší ze dvou hodnot – původně nalezeného listu a vkládaného prvku. Klíčem vnitřního uzlu v se stává hodnota jeho levého syna, pravý syn přebírá zbývající hodnotu. Příklad vložení nového prvku do stromu zachycuje obrázek 3.1.

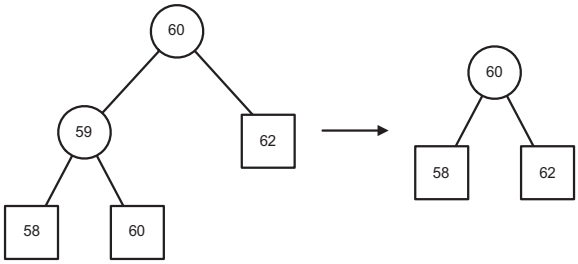


Obr. 3.1: Příklad vložení hodnoty 14 do BVS

3.3 Operace DELETE

Smazání klíče z binárního vyhledávacího stromu předchází jeho vyhledání (opět pomocí operace *member*, která je popsána v odstavci 3.1). Pokud se hodnota nalezeného listu liší od klíče, který chceme odstranit, funkce končí

neúspěchem. V opačném případě následuje smazání listu společně s jeho rodičem. Názornou ukázkou je příklad na obrázku 3.2.



Obr. 3.2: Příklad odstranění hodnoty 60 z BSV

Časová složitost všech dříve popsaných operací (*insert*, *delete* a *member*) v binárním vyhledávacím stromě T je úměrná jeho výšce. V průměrném případě bude složitost $O(\log |T|)$. Může ovšem nastat situace, kdy strom zdegeneruje na obyčejný spojový seznam a jeho výška se změní na $|T|$. Tudíž složitost v nejhorším případě je $O(|T|)$. Stromy zaručující optimální výšku $O(\log |T|)$ a tím i časovou složitost v nejhorším případě $O(\log |T|)$ se nazývají vyvážené stromy. Mezi ně patří, jak již bylo napsáno v úvodu, např. AVL stromy, červeno-černé stromy nebo B-stromy.

4 Červeno-černé stromy

V následujících kapitolách se seznámíme s jednotlivými typy červeno-černých stromů. U každé verze formulujeme definici, popis operací *insert* a *delete* a vyvažovací operace.

4.1 Standardní červeno-černé stromy

V literatuře také označovaný jako striktně vyvážený červeno-černý strom (anglicky *strictly balanced red-black tree*). Jejich uplatnění najdeme například při implementaci jiných datových struktur v knihovně jazyka C++ *STL*.

Poznámka. V textu budeme někdy používat zkratku *RB* pro označení červeno-černého stromu odvozenou z anglického spojení *red-black*.

Definice 2. Červeno-černý strom je listově orientovaný binární vyhledávací strom (viz definice 1), jehož uzly jsou obarveny červeně nebo černě, a splňuje následující podmínky:

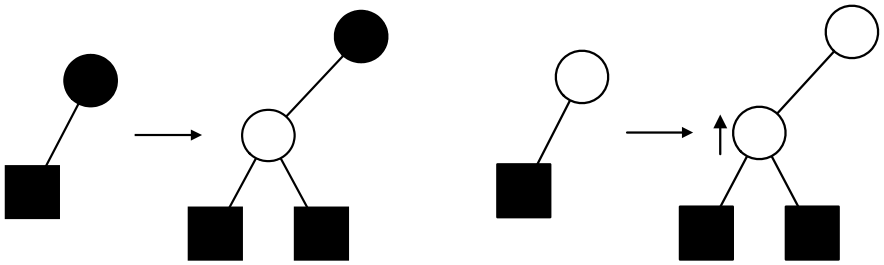
1. každá cesta z kořene stromu do listu obsahuje stejný počet černých uzlů
2. každý červený uzel (kromě kořene) má černého otce
3. všechny listy stromu jsou černé.

Věta 4.1. *Maximální výška červeno-černého stromu T je menší nebo rovna* $2 \log |T|$.

Důkaz. Viz [1] □

4.1.1 Operace INSERT

Postupujeme stejně jako u binárního vyhledávacího stromu. Následně nový vnitřní uzel (označme jej v) obarvíme červeně a jeho listy černě. Situace bezprostředně po vložení nového prvku do stromu jsou znázorněny na obrázku 4.1.



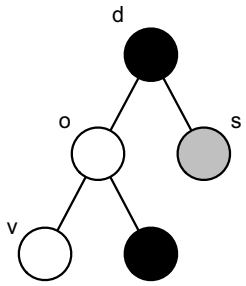
Obr. 4.1: Situace po vložení nového prvku do stromu

Poznámka. Pro všechny obrázky v diplomové práci platí následující konvence: čtverec znázorňuje list a kruh obecný uzel, tj. list nebo vnitřní uzel stromu. Vrcholy vyplněné černou barvou reprezentují černé uzly, bílé symbolizují červené vrcholy. Šedá barva vyjadřuje vrchol, který může být červený nebo černý. Symetrické případy vynecháváme.

Na obrázcích v této kapitole symbol "↑" označuje volání vyvažovací operace po vložení nového prvku do stromu, "↓" spuštění vyvažovací procedury po odstranění uzlu a "×" znázorňuje uzel, který bude smazán.

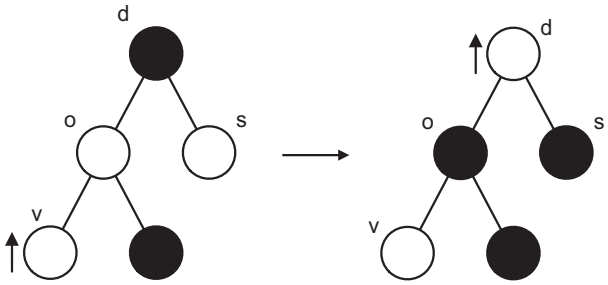
V případě, že otec nového vnitřního uzlu v je červený (stejně jako vrchol v), výsledný strom již není červeno-černý strom (došlo k porušení pravidla 2 z definice 2), viz obrázek 4.1b. Za účelem napravit tuto situaci a obnovit vyvážený stav, aplikujeme vyvažovací operaci na nově vložený vnitřní uzel. Kdykoliv je tato operace volána na uzel v, tak v je červený. Všimněme si, že pokud otec v je také červený (v našem textu označen jako o), musí být prarodič d černý (v případě existence). Cílem vyvažovací operace tudíž je, aby rodič o byl černý, zatímco počet černých uzlů na jakékoliv cestě z kořene stromu do listů se nezměnil. Nyní přecházíme k řešení vzniklého problému. Nejprve záleží na barvě uzlu s (strýc v), viz obrázek 4.2:

1. s je červený. Pak pouze přebarvíme o, s a d podle obrázku 4.3. Podmínky 1 a 3 z definice červeno-černého stromu (definice 2) jsou splněny.



Obr. 4.2: Obecná situace před vyvažováním po operaci INSERT

Bod číslo 2 může být porušen, ale kolizi jsme posunuli o 2 hladiny blíže ke kořeni stromu. Tudíž aplikujeme znovu vyvažovací operaci na uzel d.



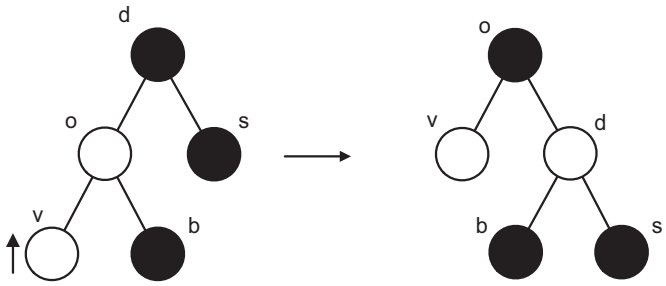
Obr. 4.3: Oprava INSERTu přebarvením

2. s je černý. Záleží na tom, zda hodnota v leží mezi hodnotami o a d nebo ne. Jinými slovy, zda cesta $v - o - d$ obsahuje zatáčku.

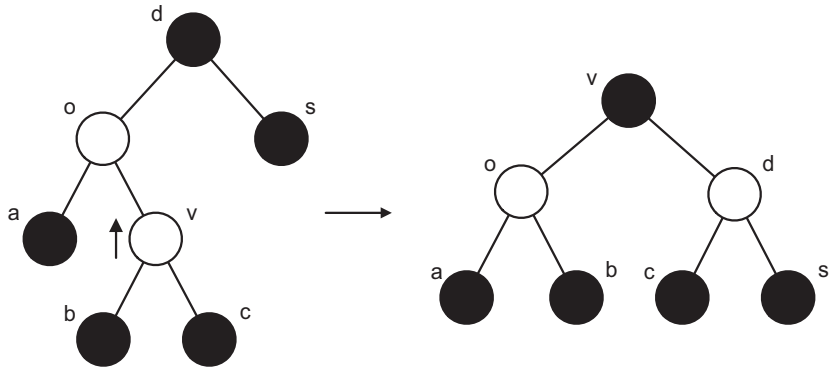
   (a) bez zatáčky – provedeme rotaci a přebarvíme podle obrázku 4.4. Budou splněny podmínky 1, 2 i 3 z definice 2 a dostali jsme tedy červeno-černý strom.

   (b) se zatáčkou – provedeme dvojitou rotaci a opět přebarvíme (obr. 4.5). Splněny jsou všechny podmínky z definice a získáme opět rovnou požadovaný červeno-černý strom.

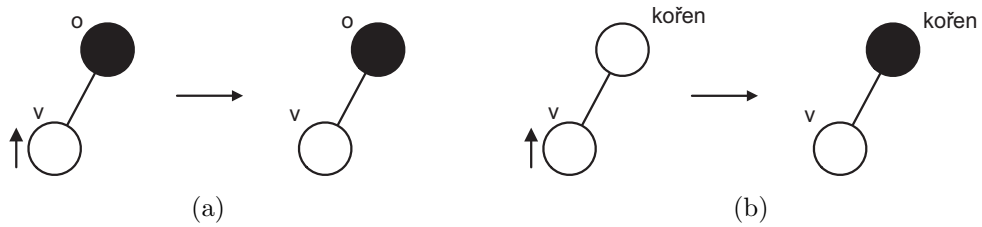
Triviální případy a jejich řešení jsou znázorněny na obrázku 4.6.



Obr. 4.4: Oprava INSERTu rotací a přebarvením



Obr. 4.5: Oprava INSERTu dvojitou rotací a přebarvením

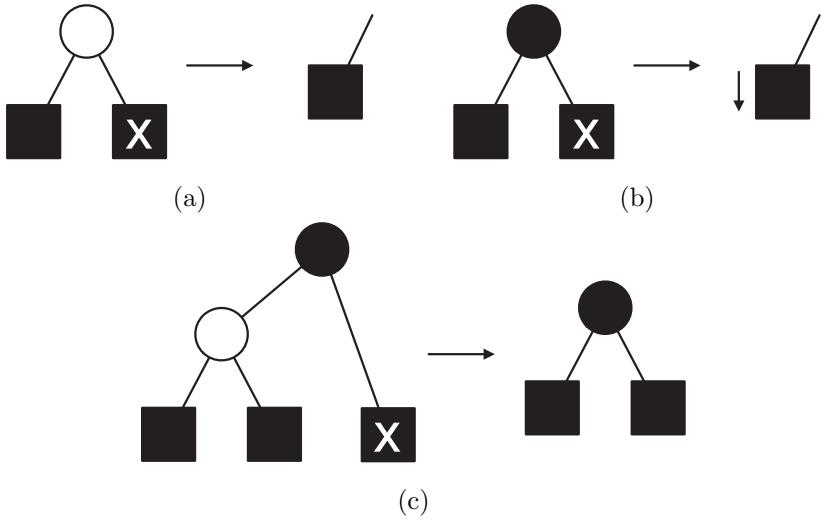


Obr. 4.6: Triviální případy po INSERTu

Protože výška RB stromu je podle věty 4.1 $O(\log |T|)$, transformace z bodu 1 se vykoná maximálně $O(\log |T|)$ krát, ostatní úpravy se provedou v čase $O(1)$. Celková složitost vložení nového prvku do RB stromu je tedy $O(\log |T|)$. Navíc se provede maximálně jedna (nebo dvojitá) rotace.

4.1.2 Operace DELETE

Budeme opět postupovat stejně jako u operace *delete* v BVS. Všimněme si, že podmínky vyváženosti (definice 2 na straně 10) implikují následující poznatek: zbývající sourozenec listu, který odstraňujeme, je také list a nebo červený uzel se dvěma listy. Pokud smazaný rodič byl černý, je narušena struktura červeno-černého stromu. Může být jednoduše opravena v případě, že zbývající vrchol je červený (pouhou změnou jeho barvy). V opačné situaci má odstranění za důsledek zavolání vyvažovací operace na zbývající uzel (obr. 4.7).



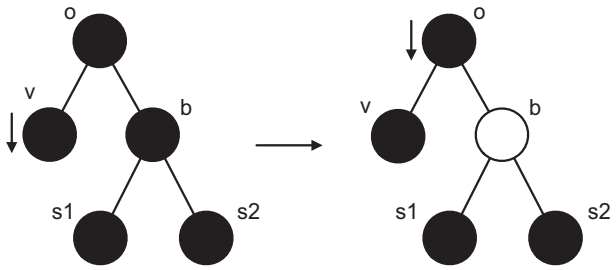
Obr. 4.7: Situace po operaci DELETE

Kdykoliv je tato vyvažovací operace aplikována na uzel v, je tento uzel černý a cesty z kořene do listů podstromu v mají o jeden černý vrchol méně než ostatní (porušena podmínka 3 z definice 2 na straně 10). Úkolem vyvažovací funkce je tedy zvýšit počet černých uzlů v podstromu v o jeden.

Úprava aktuálního stromu je závislá na barvě vrcholu b (bratr otce odstraněného listu):

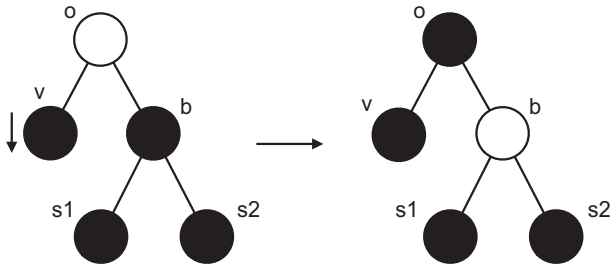
1. b je černý. Rozlišujeme dále 4 případy, z nichž jeden propaguje poruchu stromu o hladinu výš a ostatní končí červeno-černým stromem.

   (a) Otec i synovci jsou černí. Přebarvíme b na červeno, viz obrázek 4.8. Tedy kolize je posunuta o hladinu výše.



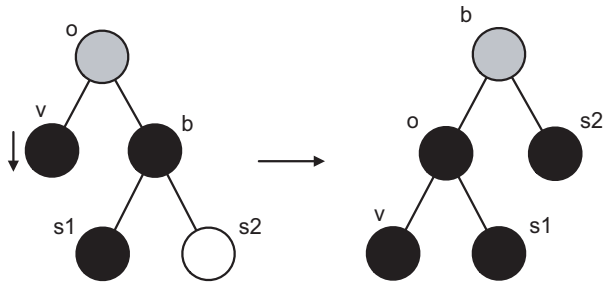
Obr. 4.8: Částečná oprava DELETE přebarvením

   (b) Otec je červený, synovci černí. Přebarvíme otce i bratra podle obrázku 4.9 a dostáváme červeno-černý strom.



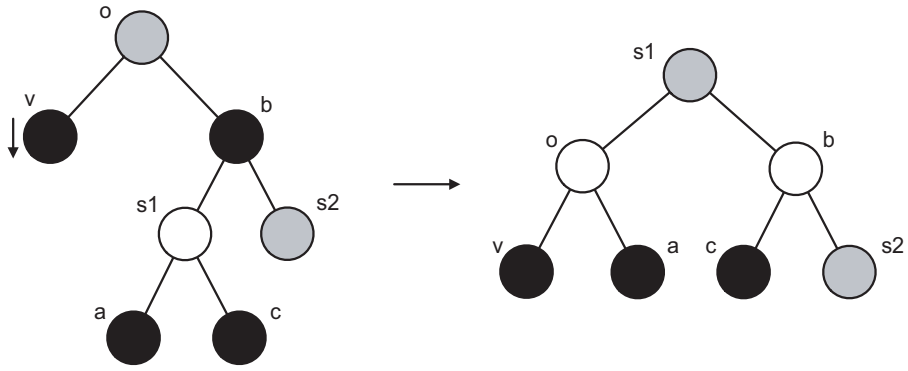
Obr. 4.9: Oprava DELETE přebarvením

(c) Synovec $s1$, jehož hodnota leží mezi hodnotami otce a bratra uzlu v, je černý, druhý synovec je červený. Přebarvíme a zrotujeme podle obrázku 4.10, tj. uzlům o a $s2$ změníme barvu na černou, vrchol b dědí původní barvu vrcholu o a ostatní barvy zůstávají nezměněny. Výsledkem je vyvážený RB strom.



Obr. 4.10: Oprava DELETE přebarvením a rotací

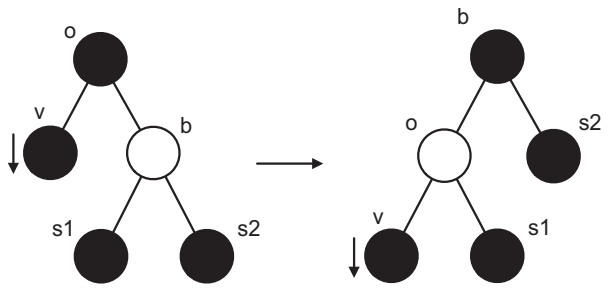
(d) Synovec $s1$ je červený, druhý synovec má libovolnou barvu. Přebarvíme (barvu vrcholů o a b změníme na červenou, uzel $s1$ přebírá původní barvu vrcholu o a zbývající vrcholy si ponechávají svoje barvy) a provedeme dvojitou rotaci. Vše zachycuje obrázek 4.11. Tato operace vede na červeno-černý strom.



Obr. 4.11: Oprava DELETE přebarvením a dvojitou rotací

2. b je červený. Provedeme rotaci. Dostaneme strom ve tvaru, který je na obrázku 4.12 a aplikujeme bod č.1. Přestože to tak na první pohled

nevypadá, jsme u konce. Bratr (uzel $s1$) je černý a otec červený, tedy příští oprava bude případ b, c nebo d z bodu 1 a tudíž skončíme červeno-černým stromem.



Obr. 4.12: Částečná oprava DELETE přebarvením a rotací

Případy b, c a d z bodu číslo 1 a situace z bodu 2 končí po provedení konstantního počtu změn barev a nejvýše dvou rotací nebo rotace a dvojité rotace, časová složitost transformace z bodu jedna je nejvýše $O(\log |T|)$. Metoda tedy pracuje s celkovou časovou složitostí $O(\log |T|)$ s provedením nejvýše dvou rotací nebo rotace a dvojité rotace.

4.2 Relaxovaně vyvážené stromy

V této části rozebereme relaxovaně vyvážený strom (viz [5, 6].), ve kterém budeme využívat stejné transformace jako v případě striktně vyváženého červeno-černého stromu popsaného v předchozí kapitole. Hlavní myšlenkou je oddělení operací *insert* a *delete* (vložení a smazání listu) od vyvažovacího algoritmu. Místo volání transformační procedury ihned po aktualizaci uložíme pouze požadavek na vyvážení konkrétního uzlu. Úprava stromu do podoby podle definice 2 je odložena a může být libovolně proložena vyhledávacími a aktualizačními operacemi. Také smazání listu izolujeme z funkce *delete* a tak výsledkem je pouze zanechání příznaku na odstranění prvku na požadovaném vrcholu. Vlastní vyjmutí klíče ze stromu je opět součástí vyvažovací úlohy.

Nyní uvedeme definici relaxovaně vyváženého červeno-černého stromu (angl. *relaxed balanced red-black tree*).

Definice 3. Relaxovaně vyvážený červeno-černý strom je binární vyhledávací strom, jehož vrcholy jsou červené nebo černé. Červené uzly mohou mít tzv. *up-in* požadavky (up-in request), černé *up-out* a listy ještě navíc mohou obsahovat požadavky na odstranění (removal request, remove request). Dále musí platit následující podmínky relaxovaného vyvážení:

1. pro každou cestu z kořene do listu platí, že součet všech černých uzlů a up-out požadavků je stejný
2. každý červený uzel (kromě kořene) má černého otce nebo up-in požadavek
3. všechny listy jsou černé.

Poznámka. *Up-in* požadavek (nebo také *push-up* ve [15]) je ekvivalentem žádosti o vyvážení po vložení prvku (na obrázcích znázorněn jako "↑"). Na takto označený uzel je nutné zavolat vyvažovací operaci, která používá stejné transformace jako vyvažovací operace u RB stromu. *Up-out* příznak (*pull-*

down ve [15]) se rovná žádosti o spuštění procesu vyvážení následujícího po aplikaci funkce *delete* (vyobrazen jako "↓").

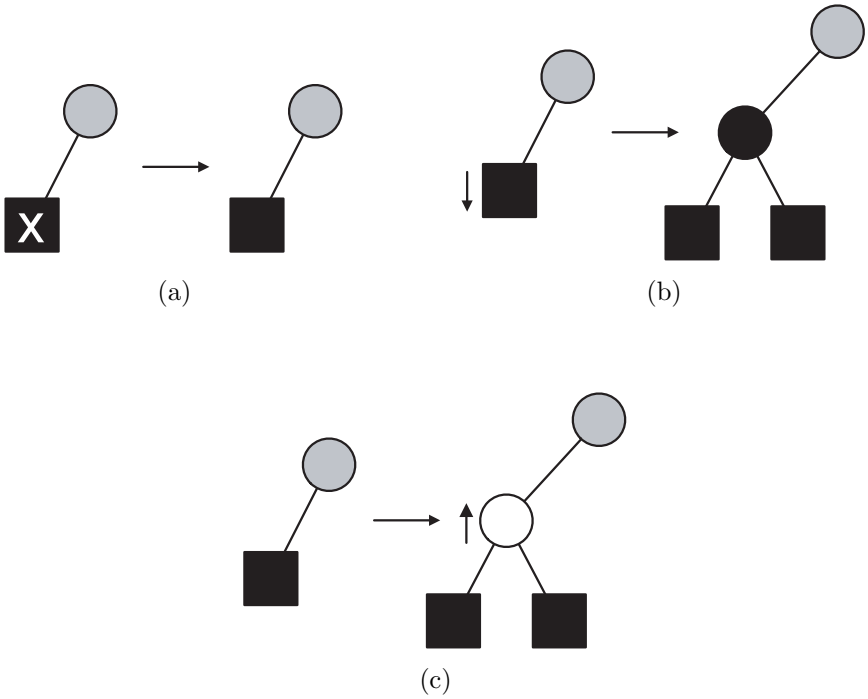
Poznámka. Na některých obrázcích v této kapitole jsou pro jednodušší popis transformací použity váhy uzlů místo barev. Hodnota 0 označuje červený uzel, 1 černý a 2 černý uzel s up-out příznakem. Tudíž je možné jedním nákresem zachytit více případů, které jsou popsány v základní verzi červeno-černého stromu několika ilustracemi. Podmínku 1 z definice 3 bychom mohli formulovat takto: pro každou cestu z kořene do listu platí, že součet hodnot všech uzlů na této cestě je stejný.

Cílem každé vyvažovací operace je, aby udržovala strom jako relaxovaně vyvážený RB strom a pozvolna jej transformovala směrem k červeno-černému stromu podle definice 2. Jak později nahlédneme, v případě akumulovaného vložení prvku do stromu lze garantovat podmínky definice 3 velmi jednoduše. Problémy ovšem mohou nastat při nahromadění několika operací *delete* pokud povolíme smazání libovolných listů společně s jejich (eventuálně černými) rodiči. Toto všechno vede k jednoduché, ale zároveň rozhodující myšlence našeho relaxovaného stromu. Namísto odstranění listu společně s jeho otcem okamžitě po nalezení ve stromě zanecháme pouze požadavek na vymazání (tzv. *removal request* nebo *remove request*, na obrázcích znázorněn jako "×"). List není fyzicky vyjmut ze stromu, ale je takto označen a je odstraněn po vyřešení eventuálních konfliktů s up-in a up-out požadavky při vyvažování.

4.2.1 Operace INSERT

Vložení nového prvku do stromu předchází nalezení jeho správné pozice mezi listy operací *member* (opět upravená verze z poznámky na konci kapitoly 3.1 jako v případě RB stromu). Pokud vyhledaný list obsahuje požadavek na odstranění, je tento příznak zrušen a hodnota je (znovu) vložena do stromu. V opačném případě nahradíme list novým vnitřním uzlem (jako v kapitole 4.1.1). Jestliže původní list obsahoval up-out žádost, zrušíme jej a vyměníme za černý vnitřní uzel se dvěma listy. Jinak barvu nastavíme na červenou a uložíme na tento uzel up-in požadavek. Nyní může nastat

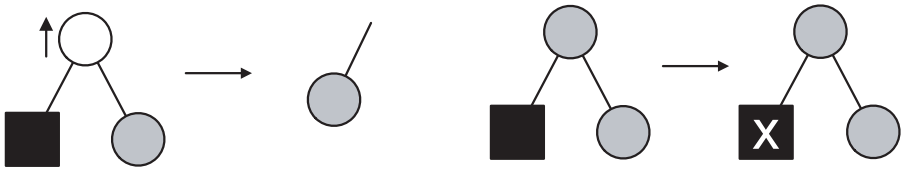
porušení podmínek definice 2. Výše popsaný proces *insert* je znázorněn na obrázcích 4.13a – 4.13c.



Obr. 4.13: INSERT

4.2.2 Operace DELETE

Před smazáním požadovaného klíče ze stromu je nutné jej opět nejdříve vyhledat. Pokud nalezený list má červeného otce s up-in příznakem (jako výsledek předchozí operace *insert*), tuto žádost zrušíme a odstraníme list společně s jeho otcem. Jinak nastavíme remove požadavek na vyhledaný list. Viz obrázky 4.14a a 4.14b.



Obr. 4.14: DELETE

4.2.3 Vyvažovací operace

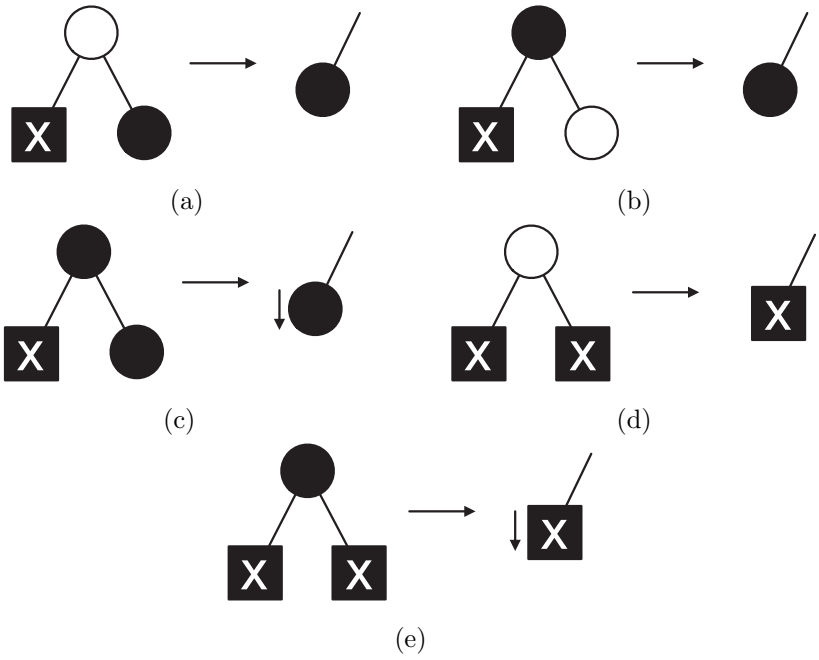
Úkolem vyvažovací operace u popisovaného typu relaxované verze červeno-černého stromu je vyřešit všechny up-in, up-out a remove požadavky. Toho může být dosaženo použitím v podstatě stejné množiny transformací jako v případě striktně vyváženého červeno-černého stromu popsaného v kapitole 4.1.

Žádost na odstranění můžeme řešit pouze v případě, že uzel, bratr ani otec neobsahují up-out příznak jako výsledek dříve uloženého požadavku na smazání listu. Provedeme tedy odstranění listu i jeho rodiče. Pokud byl otec černý a zbývající syn je červený, změníme jeho barvu na černou. Jestliže druhý syn má také remove příznak, musíme posunout tento příznak na otce. V případě černého otce a černého bratra nastavíme up-out požadavek na otce. Popsané úpravy můžeme nalézt na obrázku 4.15.

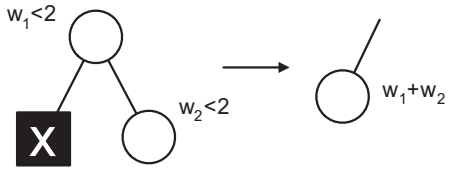
Modifikace na obrázcích 4.15a – 4.15c můžeme pomocí již dříve zmíněných hodnot uzlů shrnout do jednoho nákresu na obrázku 4.16.

Up-in nebo up-out příznaky odstraníme aplikováním jednoduchého restrukturalizačního kroku, stejně jako při řešení konfliktů ve vyváženém RB stromu (viz kapitola 4.1.1 a 4.1.2, obrázky 4.2 – 4.12). Tím je nerovnováha zrušena nebo je přesunuta na rodiče (prarodiče) a vyřešena později.

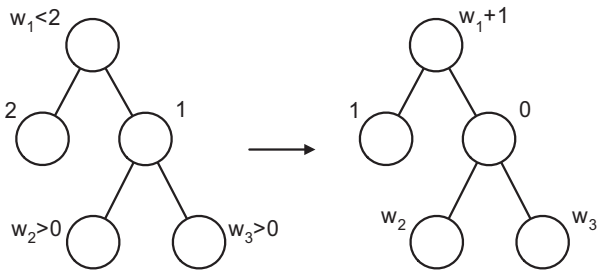
Transformace vyváženého červeno-černého stromu na obrázcích 4.8 a 4.9 lze zjednodušeně implementovat podle obrázku 4.17.



Obr. 4.15: Odstranění REMOVE požadavku

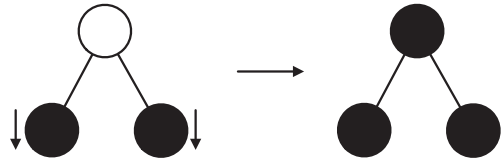


Obr. 4.16

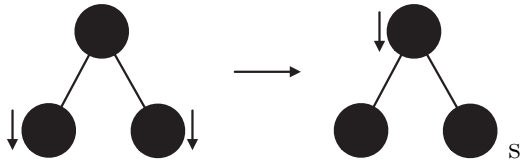


Obr. 4.17: Zjednodušení transformace

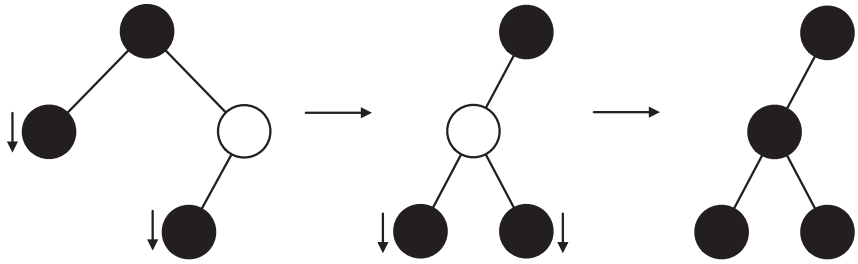
V průběhu řešení příznaků v relaxovaném RB stromu popsaných výše mohou nastat dvě odlišné situace, které se ve vyváženém červeno-černém stromu nevyskytují. Pokud oba synové mají up-out požadavek, tak je lze odstranit následující cestou: je-li otec *o* těchto dvou uzlů červený, potom uzel *o* obarvíme černou barvou a up-out požadavky z obou synů vymažeme. V případě černé barvy otce *o* opět zrušíme příznaky na obou listech a přesuneme je na uzel *o* (problém tím posuneme o hladinu výš). Viz obrázek 4.18a a 4.18b. Druhá odlišná transformace kombinuje úpravu aplikovanou na uzel s up-out požadavkem a červeným sourozencem a první přidanou operaci. Nejdříve provedeme rotaci a následně dostaneme případ číslo 1 (obr. 4.18c).



(a)



(b)



(c) Rotace a přebarvení

Obr. 4.18: Odlišné transformace oproti vyváženému RB stromu

Jestliže nelze aplikovat žádnou z uvedených transformací na uzel, který obsahuje jeden z příznaků (up-in, up-out, remove), musíme nejdříve vyřešit žádosti na uzlech v jeho bezprostředním okolí podle následujícího pořadí:

1. up-out
2. up-in
3. remove

Toto pořadí priorit vyřizování požadavků platí i pro žádosti na jednom uzlu, tj. nejdříve odstraníme up-out a následně můžeme řešit remove, který je případně také na tomto listu (případ na obrázku 4.15b).

Poznámka. Bezprostředním okolím uzlu myslíme uzly, podle kterých se rozhodujeme, jakou transformaci zvolíme, tj. otec, bratr, případně praotec, strýc a synovci.

Následující lemma nám vždy zajišťuje existenci uzlu, na který může použít jednu z transformací.

Lemma 4.2. *Uvažujme relaxovaně vyvážený červeno-černý strom, který nesplňuje podmínky definice červeno-černého stromu. Potom existuje alespoň jeden uzel, na který lze aplikovat jedna z uvedených transformací.*

Důkaz. Viz [6]. □

Věta 4.3. *Nechť T je červeno-černý strom. Uvažujme i vložení a d odstranění prvku aplikovaných na T. Potom k obnovení vyváženosti červeno-černého stromu je potřeba $O(i + d)$ strukturálních změn a $O(\log(n + i))$ přebarvení, kde n je velikost stromu T.*

Důkaz. Viz [6]. □

4.3 Chromatické stromy

Definice 4. Uzly chromatického stromu jsou červené, černé nebo přetížené (*overweight*). Barva každého uzlu v je reprezentována jako číselná hodnota $w(v)$, váha v. Váha červeného uzlu je nula, černého jedna a přetížený uzel má hodnotu váhy větší než jedna. Chromatický strom dále musí splňovat následující podmínky:

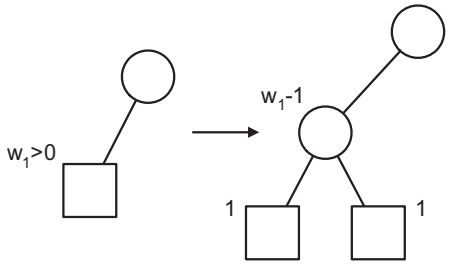
1. žádný list není červený (tj. $w(v) > 0$)
2. součet vah uzlů na každé cestě z kořene do listu je stejný.

Poznámka. Tato definice je uvedena v [5, 9]. Trochu rozdílný pohled na chromatické stromy popisují publikace [2, 12, 13], kde váhu přiřazují hranám stromu, nikoli uzlům.

Operace vložení a vymazání prvku v chromatickém stromě jsou řešeny následujícími způsoby.

4.3.1 Operace INSERT

Nalezneme příslušné místo pro nový list (stejně jako v obou předchozích případech popisovaných stromů). Výsledný list nahradíme vnitřním uzlem a nastavíme jeho váhu jako hodnotu váhy původního listu minus jedna. Novým listům přiřadíme váhu 1. Tato situace je znázorněna na obrázku 4.19.



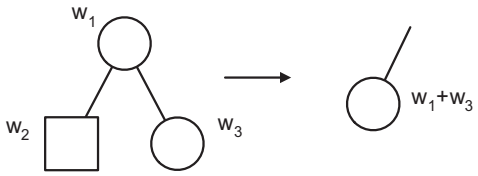
Obr. 4.19: Operace INSERT

Poznámka. Čísla a jmenovky na obrázcích v této kapitole označují váhy uzlů: 0 = červený, 1 = černý a > 1 je přetížený, přesně podle definice 4. Pokud

není u váhy uzlu naznačena jeho hodnota (např. $w = 1$ nebo pouze 1), potom je pro konkrétní transformaci nepodstatná.

4.3.2 Operace DELETE

Opět smažeme vyhledaný list společně s jeho otcem a váhu zbývajícího listu (uzlu) zvýšíme o váhu původního otce (obr. 4.20).



Obr. 4.20: Operace DELETE

4.3.3 Vyvažovací operace

Chromatický strom může mít dva typy anomálií, které mu brání být červeno-černým stromem. Výsledkem operace vložení prvku mohou být dva sousední červené uzly na cestě z kořene do nového listu.

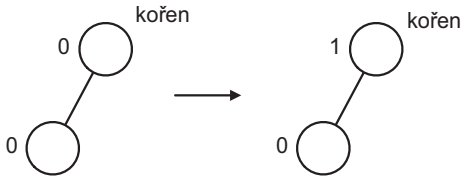
Definice 5. Říkáme, že v uzlu v je *red-red konflikt*, pokud v je červený a má červeného otce.

Odstranění klíče ze stromu naopak může produkovat druhou odchylku, přetížené uzly. Tento jev se nazývá *overweight konflikt*.

V případě, že chromatický strom neobsahuje žádné red-red a overweight konflikty, splňuje podmínky vyváženosti standardního červeno-černého stromu. Cílem vyvažovací operace chromatického stromu je tudíž odstranit všechny tyto kolize.

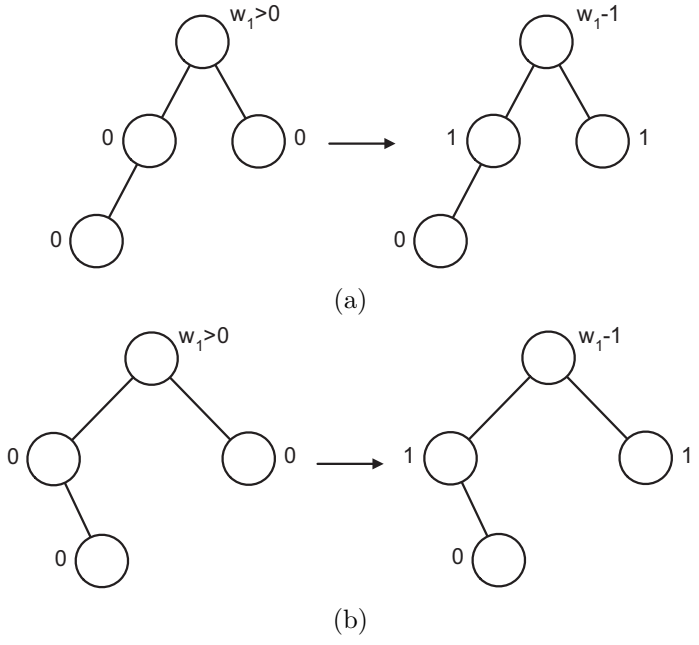
Nyní uvedeme jednotlivé operace, které odstraňují výše popsané nerovnováhy. Existují následující postupy řešící red-red konflikt:

1. triviální situaci v kořeni stromu vyřešíme jednoduchou změnou barev (viz obrázek 4.21)



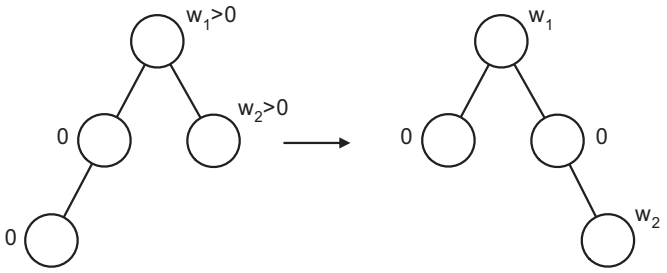
Obr. 4.21: RED-ROOT

2. *blacking* transformace: řeší red-red konflikt pouze využitím změny barev. Kolize není zrušena, ale operace ji propaguje o dvě hladiny výše (přenesením na praotce). Postup je znázorněn na obrázku 4.22.

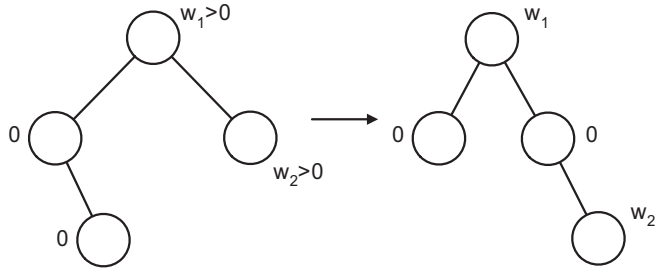


Obr. 4.22: BLACKING

3. *red-balancing* transformace: odstraňují red-red konflikt pomocí jednoduché rotace nebo dvojité rotace. Viz obrázek 4.23a a 4.23b.



(a) RED-BALANCING – jednoduchá rotace

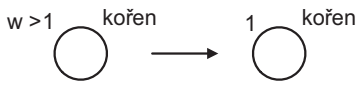


(b) RED-BALANCING – dvojitá rotace

Obr. 4.23

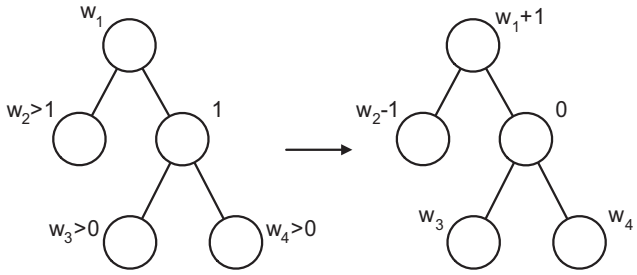
Pro odstranění přetížení máme k dispozici následující dva druhy metod a jednu triviální úpravu:

1. zrušení overweigth konfliktu v kořeni stromu je znázorněno na obrázku 4.24.



Obr. 4.24: Triviální úprava

2. *push* transformace (viz obrázek 4.25): řeší přetížení přebarvením. Konflikt může být plně odstraněn nebo přenesen o úroveň výš (na otce).



Obr. 4.25: PUSH

3. *weight-decreasing* transformace: zruší kolizi pomocí strukturální změny (nejvýše dvě rotace nebo rotace a dvojitá rotace). Všechny případy jsou zachyceny na obrázcích 4.26a – 4.26g.

Poznámka. Pokud ve výše popsaných transformacích uvádíme, že přetížení je odstraněno, myslíme tím případ uzlu s váhou 2. U větší váhy dojde pouze ke snížení hodnoty přetížení, ale konflikt není plně vyřešen. Musíme tedy znovu (několikrát) aplikovat vyvážení.

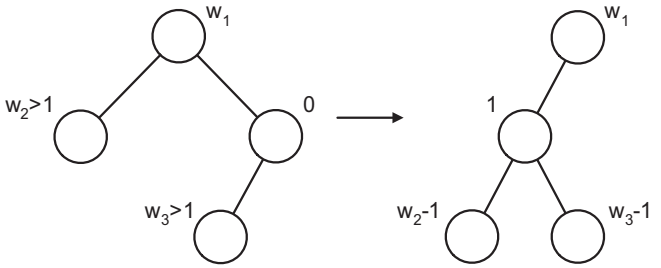
Nyní si uvedeme několik vět, které dokazují odhad počtu provedených operací. Tudíž po konečném počtu aplikovaných operací v libovolném pořadí se z chromatického stromu stává červeno-černý strom.

Věta 4.4. *Uvažujme chromatický strom T, který nesplňuje podmínky vyváženosti pro červeno-černý strom. Potom existuje alespoň jedna vyvažovací transformace z kapitoly 4.3.3, kterou můžeme aplikovat na strom T.*

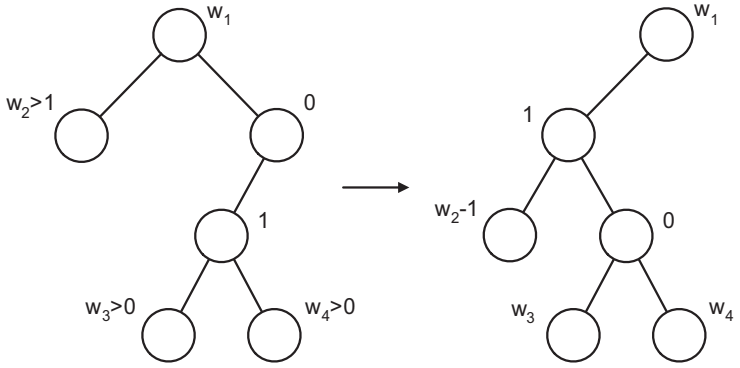
Důkaz. Viz [2, 8, 9, 12, 13]. □

Věta 4.5. *Mějme chromatický strom T. Libovolná dostatečně dlouhá sekvence vyvažovacích transformací modifikuje T do podoby červeno-černého stromu.*

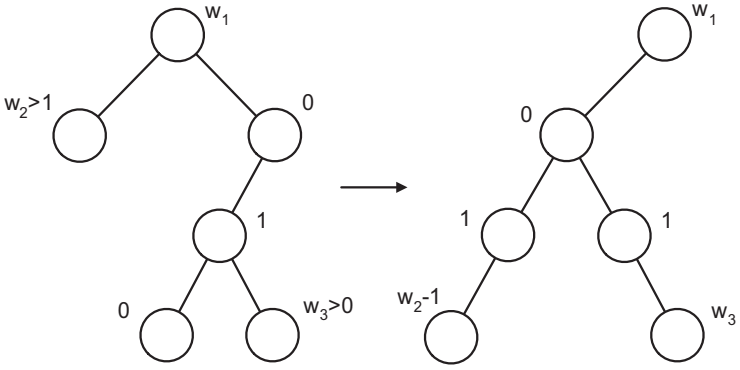
Důkaz. Viz [12, 13] □



(a) rotace

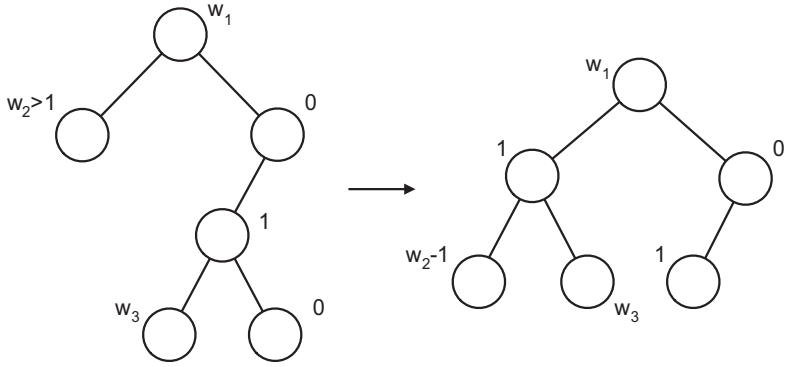


(b) rotace

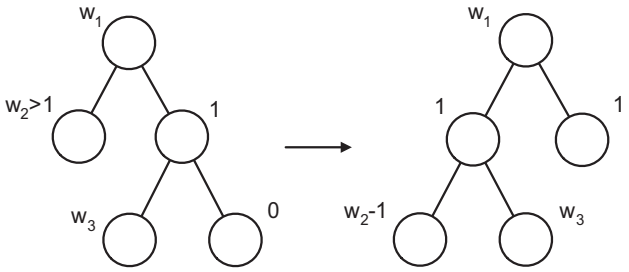


(c) rotace a dvojitá rotace

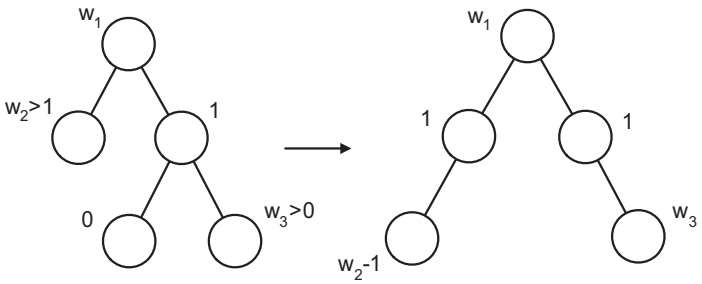
Obr. 4.26: WEIGHT-DECREASING



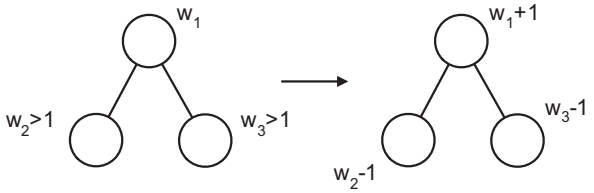
(d) dvě rotace



(e) rotace



(f) dvojitá rotace



(g) změna vah

Obr. 4.26: WEIGHT-DECREASING

Věta 4.6. *Pokud T je chromatický strom o velikosti n, který splňuje vyvažovací podmínky standardního červeno-černého stromu, a i vložení a d odstranění prvku je aplikováno na T, potom $O(i \cdot \log(n+i))$ blacking transformací, $O(i)$ red-balancing transformací, $O(d \cdot \log(n+i))$ push transformací a $O(d)$ weight-decreasing transformací je nutných pro obnovení vyváženosti standardního červeno-černého stromu.*

Důkaz. Viz [2, 8] □

4.4 Chromatické stromy – typ 2

Tento typ relaxovaného stromu nemá jednoznačně definovaný název. V některých literaturách (např. [5, 8]) je označen jako *relaxed red-black tree*, tedy stejně jako naše první relaxovaná struktura z kapitoly 4.2, v jiných pouze *chromatic tree* ([12, 13]). V této práci budeme používat označení chromatický strom typu 2.

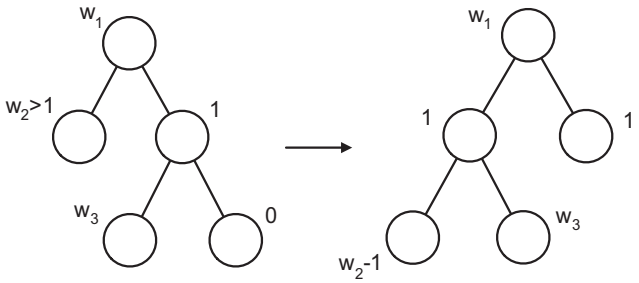
Definice chromatického stromu typu 2, operace *insert* a *delete* a transformace odstraňující red-red konflikt jsou totožné s chromatickým stromem popsaným v předchozí sekci (kapitola 4.3), tedy typem 1. Množina operací, které jsou aplikovány na overweight konflikt, je zredukována na menší kolekci transformací, která potřebuje nejvýše jednu rotaci nebo dvojitou rotaci. Obsahuje následující operace:

1. *Weight-push* transformace používá pouze přebarvení. Konflikt je odstraněn nebo přesunut o hladinu výš (na otce). Operace je shodná s *push* úpravou v chromatickém stromě na obrázku 4.25 na straně 29.

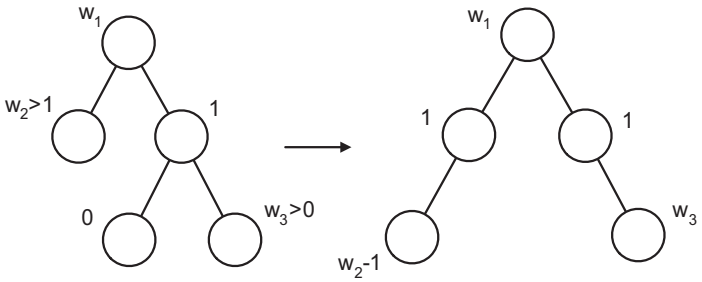
2. *Weight-decreasing* transformace odstraňuje přetížení pomocí změny struktury stromu (nejvýše jedna jednoduchá rotace nebo jedna dvojitá rotace). Obrázek 4.27a – 4.27c.

3. *Weight-temp* transformace provádí rotaci jako přípravu na spuštění weight-decreasing operace, ale sama konflikt neřeší. Úprava je znázorněna na obrázku 4.28.

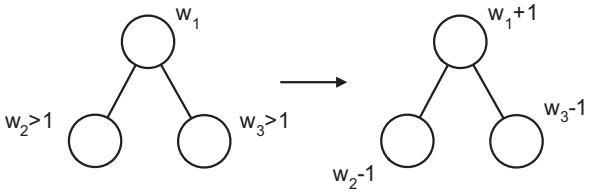
Poznámka. Protože transformace weight-temp je oddělena od operací, které po ní následují, umožňuje na rozdíl od chromatického stromu typu 1 větší míru souběžnosti několika procesů v paralelním prostředí.



(a) rotace

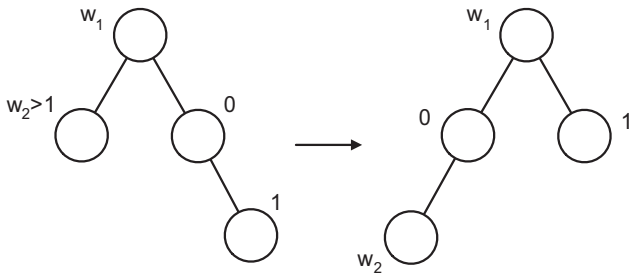


(b) dvojitá rotace



(c) změna vah

Obr. 4.27: WEIGHT-DECREASING



Obr. 4.28: WEIGHT-TEMP (rotace)

4.5 Porovnání relaxovaných algoritmů

V této kapitole porovnáme vyvažovací operace všech tří dříve popsaných relaxovaně vyvážených algoritmů z hlediska jejich hlavních odlišností a podobností.

Poznámka. Čísla v tabulkách v této kapitole odkazují na čísla příslušných obrázků k jednotlivým transformacím.

4.5.1 INSERT vyvažování

Nejprve porovnáme transformace odstraňující nevyváženosti způsobené vložením nového prvku do stromu. Rozdíl mezi relaxovaným a oběma typy chromatických stromů je v tom, že relaxovaný strom označuje konfliktní uzly up-in požadavky a chromatické stromy řeší red-red kolize. Pokud se ovšem podrobněji podíváme na obě situace, zjistíme, že up-in požadavek vlastně označuje červený uzel, který ma červeného otce, tj. přesně red-red konflikt v chromatickém stromu. Tudíž vyvažovací operace všech tří námi popsaných verzí upravených červeno-černých stromů řeší stejný problém – dva po sobě jdoucí červené vrcholy.

Jak lze dále snadno nahlédnout, vyvažovací operace jsou pro všechny typy červeno-černých stromů shodné. V tabulce 4.1 je přehledně zobrazena předchozí úvaha. Čísla v jednotlivých řádcích korespondují s obrázky znázorňujícími tyto operace. V posledním sloupci tabulky najdeme popis změn, které jsou prováděny během transformací.

Červeno-černý a relaxovaný	Chromatický typ 1 a 2	Operace
4.3	BLACKING 4.22	přebarvení
4.6	RED-ROOT 4.21	přebarvení
4.4	RED-BALANCING 4.23a	rotace
4.5	RED-BALANCING 4.23b	dvojitá rotace

Tab. 4.1: Porovnání vyvažovacích transformací po operaci insert

4.5.2 DELETE vyvažování

Při porovnávání vyvažovacích operací volaných po operaci *delete* nastává poněkud složitější situace než v předešlém případě. Chromatické stromy totiž, na rozdíl od relaxovaného stromu, odstraňují uzel ihned ve funkci *delete*, zatímco v relaxovaném stromu je odstranění součástí vyvažovací metody. Protože navíc chromatický strom podporuje akumulovaný *delete* ve formě vícenásobného přetížení uzlu, bylo nutné zavést u relaxovaného RB stromu požadavek na odstranění (*removal request*).

Pokud se podíváme do tabulky č. 4.2 zjistíme, že operace v jednotlivých typech stromů mají své ekvivalenty i v ostatních strukturách.

Červeno-černý	Relaxovaný	Chromatický 1	Chromatický 2	Transformace
4.12 (a)	4.12 (a)	první část (d), (f) a (g)	TEMP 4.28	rotace
4.8 (b), 4.9 (c)	4.17 (b)	PUSH 4.25	PUSH 4.25	přebarvení
4.10 (d)	4.10 (c)	4.26e (a)	4.27a (a)	rotace
4.11 (e)	4.11 (d)	4.26f (b)	4.27b (b)	dvojitá rotace
—	4.18a, 4.18b (e)	4.26g (c)	4.27c (c)	přebarvení
(a)+(c)	(a)+(b)	4.26b (d)	TEMP+PUSH	rotace
(a)+(d)	(a)+(c)	4.26d (e)	TEMP+(a)	2 rotace
(a)+(e)	(a)+(d)	4.26c (f)	TEMP+(b)	rotace+dvojitá rotace
—	4.18c (a)+(e)	4.26a (g)	TEMP+(c)	rotace

Tab. 4.2: Porovnání vyvažovacích transformací po operaci delete

Tímto podrobným porovnáním jsme zjistili, že tři námi popsané relaxované verze RB stromů jsou velmi podobné. Chromatické stromy mohou být podle [5] považovány za zobecnění relaxovaného vyváženého RB stromu, ve kterém bychom povolili akumulování více up-out požadavků v jenom uzlu.

4.6 Administrace vyvažovacích požadavků

V této kapitole se zamyslíme nad doplňkovým algoritmem, jenž je nezbytný pro implementaci vyvažovacího procesu v relaxované verzi červeno-černých stromů.

Pokud je v relaxovaně vyváženém stromu prováděna aktualizace, může mít několik uzlů ve stromě narušenu podmínku vyváženého stavu. Za účelem označení takovýchto míst ukládá aktualizační operace v příslušných uzlech požadavky na vyvážení. To napomáhá vyvažovacímu procesu najít nerovnovážný stav a rozhodnout, která transformace by měla být použita pro znovu nastolení vyváženého stavu.

V literatuře se objevují tři rozdílné návrhy detekce požadavků vyvažovacím procesem. Jednou z možností je náhodně procházet datovou strukturu a vyhledávat žádosti (viz [12, 13]). V každém kroku je prohlížena malá část (začínající nejvyšším uzlem ve směru shora dolů) a je zkoumáno, zda aplikovat transformaci.

Pravidelný vyvažovací algoritmus je prezentován v [11]. Během vkládání a mazání jsou všechny uzly na cestě od kořene až po upravovaný list označeny. Tím vyvažovací proces přesně zná podstromy obsahující kolize s definicí RB stromu a omezuje se při hledání požadavků pouze na tyto označené části stromu.

Třetím návrhem je použití tzv. problémové fronty (*problem queue*), která je popsána v [1, 2, 8]. Kdykoli je vytvořena nerovnováha, vložíme vrchol, který způsobuje tuto "nevyváženost", do fronty vyvažovací funkce.

Problémová fronta V následujícím textu si podrobněji rozebereme myšlenku a implementaci problémové fronty, která je součástí naší implementace relaxovaných verzí červeno-černého stromu.

Jak již bylo popsáno výše, kdykoli je vygenerován požadavek na vyvážení, uložíme odkaz na příslušný uzel do fronty. Tudíž každá žádost je reprezentována prvkem v problémové frontě. Pokud je prázdná (respektive všechny problémové fronty jsou prázdné, v případě použití více front), je strom vy-

vážený. Může ovšem nastat situace, kdy se vyvažovací požadavek vyřeší automaticky aktualizační nebo vyvažovací metodou aplikovanou na jinou žádost. Tudíž potom fronta obsahuje i ukazatele na uzly, které již neobsahují příznak pro vyvážení nebo dokonce již nejsou součástí stromu (mohly být odstraněny). Tyto krajní případy řešíme testem na existenci uzlu nebo požadavku na něm.

Pro zjednodušení budeme dále požadovat, aby každý uzel byl reprezentován ve frontě maximálně jedním odkazem. Vytvoříme v každém vrcholu novou proměnnou, která bude nastavena (např. na *true*) v případě, že vložíme odkaz na uzel do problémové fronty. A zneplatněna (např. nastavena na *false*) pokud je prvek z fronty odstraněn. Tím zamezíme vícenásobnému vložení odkazu jednoho vrcholu do fronty.

4.7 Experimenty v paralelním prostředí

V mnoha publikacích můžeme nalézt popis relaxovaných verzí červeno-černých stromů z hlediska využití a chování v paralelním prostředí. Zaměření této práce je ovšem poněkud odlišné, a proto se v této kapitole pokusíme pouze interpretovat a shrnout výsledky testů provedených v [5].

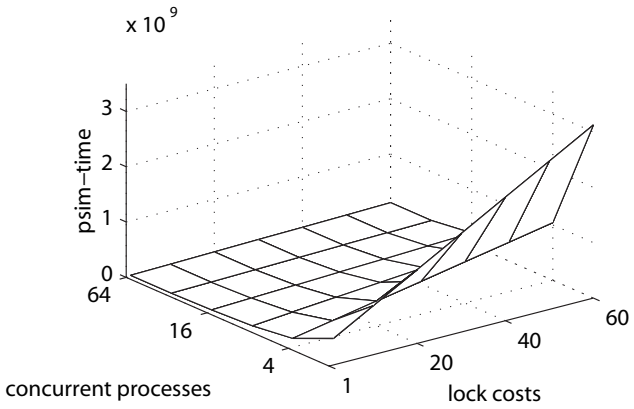
Cílem studie bylo zkoumat rozdíly tří námi popsaných relaxovaných verzí RB stromu a standardního červeno-černého stromu v paralelním prostředí. K tomu účelu autoři využili simulátor *psim*, který simuluje abstraktní model multiprocesorového stroje. Dále v algoritmech implementovali uzamykací metodu popsanou Nurmim a Soisalon-Soininenem v [12, 13] používající tři druhy zámků (r-lock, w-lock a x-lock). Uzamykací algoritmus je nedílnou součástí programu kvůli vyloučení konfliktů a k zajištění správného vykonávání programu v prostředí s více procesy.

Testovací úlohy obsahovaly 1000000 tzv. slovníkových operací, tj. vložení (*insert*), smazání (*delete*) a vyhledání (*search*) prvku, s pravděpodobnostmi výskytu $p_{insert} = 0,5$, $p_{delete} = 0,2$ a $p_{search} = 0,3$. Tyto transformace byly aplikovány na předem inicializovaný vyvážený červeno-černý strom, který obsahoval 1000000 náhodně vygenerovaných hodnot. Každý experiment provedli pomocí 1, 3, 7, 15 a 31 souběžně spuštěných procesů. Vyvažování zajišťovala metoda běžící na pozadí ostatních uživatelských procesů.

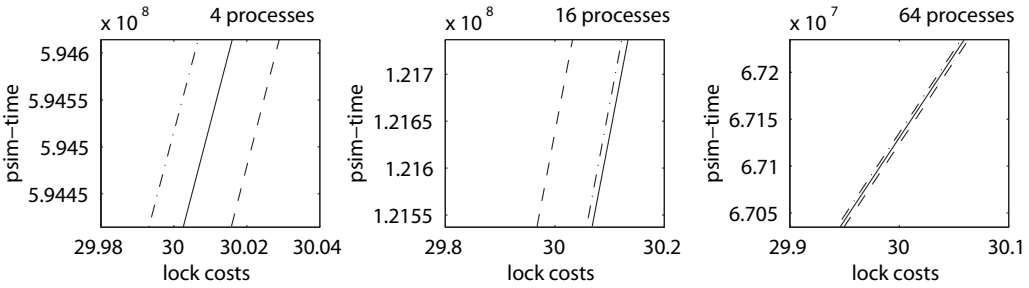
Na obrázku 4.29a můžeme vidět graf výsledku jednoho z testů znázorňující celkový průměrný čas potřebný k provedení 1000000 slovníkových operací (podle popisu v předchozím odstavci) všech tří relaxovaných verzí červeno-černého stromu v závislosti na počtu paralelních procesů a na ceně zámku[1]. Výsledky jednotlivých algoritmů jsou velice podobné a jejich rozdíly lze vypozorovat ze zvětšeného grafu na obrázku 4.29b.

Grafy 4.30a – 4.30c a 4.31 zachycují výsledky experimentů porovnávajících průměrné časy potřebné k provedení jednotlivých operací *insert*, *delete* a *member* a celkový průměrný čas 1000000 těchto operací na standardním

[1]Čas potřebný k provedení operace uzamčení uzlu stromu. Hodnota zvoleného rozmezí $1 - 60$ byla experimentálně změřena. Podrobnější popis se nachází ve [5] na straně 13.

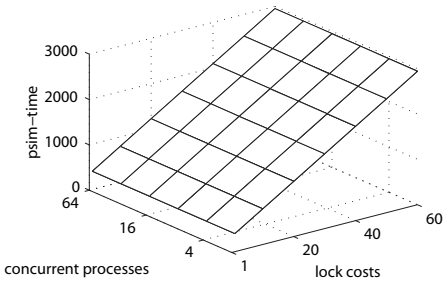


(a) Celkový graf

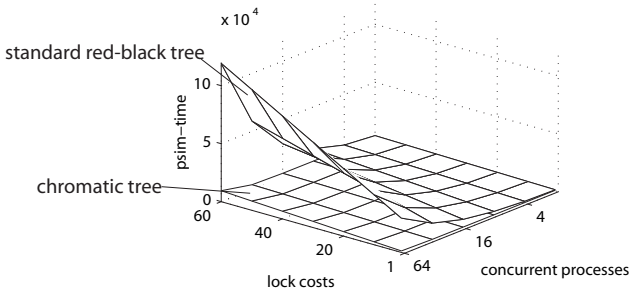


(b) Zvětšení

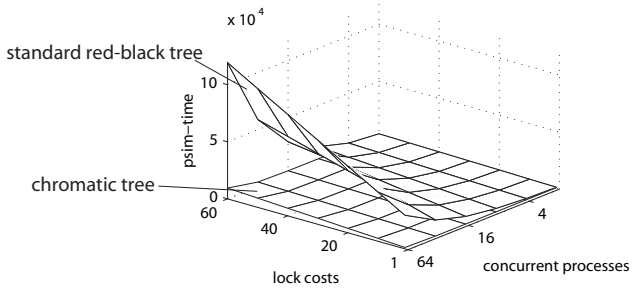
Obr. 4.29: Celkový průměrný čas potřebný k provedení 1000000 slovníkových operací. Legenda: – – relaxovaně vyvážený červeno-černý strom, — chromatický strom, – · – chromatický strom typ 2



(a) member



(b) insert

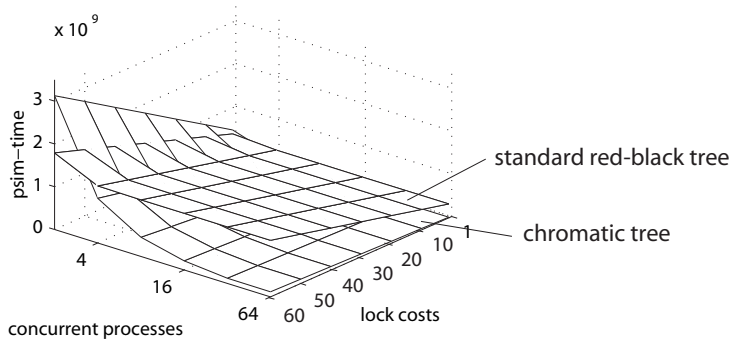


(c) delete

Obr. 4.30: Srovnání standardního červeno-černého stromu a chromatického. Průměrný čas potřebný k provedení jedné slovníkové operace.

červeno-černém stromu a na chromatickém stromu, který byl vybrán jako reprezentant relaxovaných verzí. Můžeme z nich snadno vyčíst, že pouze metoda vyhledání prvku v RB stromu dosahuje srovnatelných výsledků jako v případě chromatického stromu. Funkce *insert* a *delete* jsou podle těchto testů znatelně výhodnější v podání relaxované verze.

Pouze v případě dvou souběžných procesů při srovnání celkového naměřeného času skončí 1000000 operací na standardním červeno-černém stromu dříve než u chromatického stromu (obr. 4.31).



Obr. 4.31: Celkový průměrný čas potřebný k vykonání 1000000 operací.

Závěry všech testů provedených ve [5] lze shrnout jednou větou z této studie[2]:

> *Výsledky experimentů potvrdily domněnku, že v paralelním prostředí má smysl používat relaxované vyvažování místo striktního.*

[2]Ve [5] na straně 24.

5 Experimentální analýza

Našim cílem bylo provádět experimenty, měřit čas trvání těchto testů a shromažďovat a analyzovat výsledky. Abych mohli snadno a rychle připravovat tato data, rozhodli jsme se vytvořit jednoduchý skriptovací jazyk pro konfiguraci experimentů, který si popíšeme v následující kapitole, a oddělit generování těchto vstupních dat od samostatného provádění testů. Výhoda je také ta, že testováni není ovlivněno přípravou dat.

Pokud budeme provádět porovnání jednotlivých datových struktur na základě časů naměřených při experimentech, musíme si nejdříve určit, jakým způsobem bude měření probíhat. Existuje celá řada více či méně přesných a vhodných prostředků pro zjišťování celkového spotřebovaného času. Jedním z nich, který je ovšem nejen pro naše účely krajně nepřijatelný a jehož použití jsme již předem zavrhli, je ruční měření například pomocí stopek. Dále se nabízí prakticky dvě přípustné možnosti. Využít nástrojů časovače poskytovaných většinou běžných operačních systémů a nebo speciálních čítačů, které máme k dispozici u některých procesorů. Pro úlohy trvající řádově milisekundy a kratší můžeme použít druhou z uvedených možností, kdy měřený čas vyčteme z čítačů procesoru, jejichž hodnota je inkrementována při každém tiku procesoru. Tento typ měření je velice ovlivněn zatížením procesoru jinými procesy, protože hodnota čítačů není vztažena na konkrétní proces, ale na celý systém. Jak později uvidíme, doba běhu našich experimentů dosahuje až desítek sekund, proto budeme při svých měřeních využívat dostupné prostředky operačního systému pro práci s časovačem. Ke zjištění příslušných informací použijeme funkci `clock()`, která vrací aktuální čas spotřebovaný procesem, z něhož je volána. Tento typ měření v sobě skrývá jednu nevýhodu. Do získané hodnoty se započítává i režie použitá vlastní funkcí `clock()`. Při délce trvání našich experimentů je toto přidané časové kvantum zanedbatelné.

Zpracování výsledků a výstupů z programu bylo realizováno vlastními shellovskými skripty a následná statistická měření a grafy pomocí programů *STATISTICA* a *Microsoft Excel*.

Experimenty jsme prováděli na počítači následujících parametrů:

CPU	AMD Duron 1900MHz
RAM	756MB DDR 400MHz
OS	Debian Etch
kernel	2.6.15-1-k7
kompilátor	g++ 4.0.3

5.1 Generování testů

Pro vytváření vstupních dat do experimentů byl v programu implementován jednoduchý generátor operací a celočíselných klíčů využívající funkci `rand()` jazyka C. Tento generátor lze konfigurovat pomocí externích souborů používající jednoduchý konfigurační jazyk integrovaný v aplikaci. Obsahuje sice malou, ale pro naše testy dostačující množinu použitelných klíčových slov, která si později popíšeme. Syntaxe jazyka je opět velmi snadná, aby umožňovala co nejméně náročné psaní skriptů pro jednotlivá měření.

Vstupní soubor se skládá z volitelných globálních nastavení a bloků definujících generované části dat. Mezi obecné funkce patří `SEED` (inicializace náhodného generátoru funkce `rand()`) a `RAND_MAX` (maximální velikost náhodného čísla). Tyto definice se nesmí objevit uvnitř cyklu, který se v našem jazyce ohraničuje pomocí konstrukce `BEGIN` a `END`. Pro počáteční nastavení generátoru je v případě vícenásobného výskytu globální proměnné vždy rozhodující její poslední uvedená hodnota. Každá definice, kromě klíčových slov `BEGIN` a `END`, se skládá ze dvou částí: proměnná + hodnota.

Druhou množinu tvoří slova vyskytující se v těle bloku:

`OPERATIONS`	celkový počet operací v daném bloku
`INSERT`, `DELETE`, `FIND`	počet operací (v % z celkového množství)
`REBALANCE`*	vyvážení
`REB_PERIOD`	pravidelné vyvažování po zadaném počtu operací

START_TIMER*	začátek měření času experimentu
STOP_TIMER*	ukončení měření času experimentu
START_STATISTIC*	začátek měření počtu prováděných operací
CLR_REB*	vymazání času spotřebovaného na vyvažování

Hodnoty jednotlivých deklarací určují číslo operace, po níž bude následovat provedení dané funkce. Některé výrazy (ve výčtu výše v textu označeny symbolem *) mohou být zadány záporným číslem, což zaručuje spuštění tohoto příkazu před operací s pořadovým číslem rovným absolutní hodnotě definice. Lze je také použít v bloku bez definovaného celkového počtu operací.

Na závěr popisu generátoru si uvedeme jednoduchý příklad včetně doplňujících komentářů, který lépe osvětlí význam a použití jednotlivých příkazů našeho jazyka. Znak # slouží pro označení začátku komentáře.

```
# globální nastavení
RAND_MAX        10000000
SEED            126

BEGIN                           # začátek 1. bloku skriptu
   OPERATIONS       1000000     # celkový počet operací v prvním bloku
   INSERT           60          # 60% operací bude vložení nového prvku
   FIND             40          # 40% operací bude vyhledání prvku
   START_TIMER      500000      # po polovině operací začneme měřit čas
   REBALANCE        -500001     # před druhou polovinou vyvážíme
END                             # konec 1. bloku

# 2. blok
BEGIN
   OPERATIONS       100000
   INSERT           60
   DELETE           40
   REB_PERIOD       500         # vyvažujeme po každém pětistém kroku
END

# 3. blok
BEGIN
   STOP_TIMER       1           # ukončení měření času
END
```

Na dříve formulovaných strukturách budeme provádět měření celkového času běhu programu v závislosti na různých vstupních datech, počítat prová-

děné transformace a měřit, kolik času z celkové doby výpočtu procesu zabírá vyvažovací operace. Výsledky experimentů byly získány spuštěním jednotlivých testů na deseti různých datech vygenerovaných pomocí odlišného nastavení náhodného generátoru (hodnota definice `SEED` v konfiguračním souboru). Každé měření jsme provedli dvakrát. V průběhu experimentu je testovaná datová struktura udržována v souladu s její definicí (tj. např. chromatický strom podle definice 4) a do podoby červeno-černého stromu je transformována vždy při zavolání vyvažovací operace po zadaném počtu operací podle definice v konfiguračním skriptu, který je vstupem pro vygenerování testovacích dat. Toto samozřejmě neplatí pro standardní červeno-černý strom, jehož experimenty z pochopitelných důvodů neobsahují volání vyvažovací metody.

Dále si podrobně popíšeme všechny prováděné experimenty, uvedeme některé jejich grafy zachycující naměřené hodnoty a rozebereme výsledky z nich vyplývající. Část výsledků je uvedena v příloze na konci diplomové práce. Zbývající tabulky, grafy, konfigurační soubory, výstupy z testů a pomocné skripty jsou umístěny na přiloženém CD, které obsahuje také zdrojové kódy programu a k nim vygenerovanou dokumentaci pomocí dokumentačního nástroje *Doxygen*.

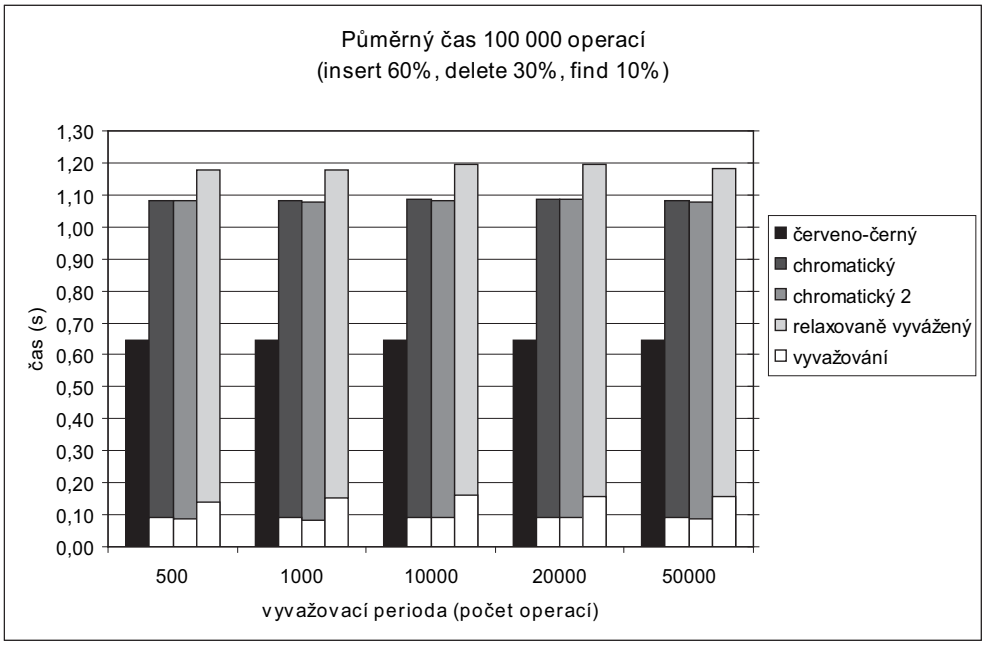
5.2 Obecný test

Jako první experiment jsme si zvolili měření celkového času potřebného k provedení zadaného počtu operací *insert*, *delete* a *find* na předem inicializovaném vyváženém červeno-černém stromu o velikosti 1000000 prvků. Na takto vytvořený strom jsme postupně aplikovali 100000, 500000, 1000000, 2000000, 4000000 a 7000000 operací v poměru 6 : 3 : 1 (insert : delete : find) s vyvažovací periodou 500, 1000, 10000, 20000 nebo 50000. Test byl zaměřen na chování jednotlivých popisovaných struktur v obecném prostředí, kde převažuje vkládání nových hodnot. Vstupní konfigurační skript pro generování dat vypadá následovně:

```
# globální nastavení
RAND_MAX        20000000
SEED            12

# vytvoření počátečního stromu
BEGIN
   OPERATIONS       1000000
   INSERT           100
   REBALANCE        1000000
END

# příprava proměnných pro testování
BEGIN
   CLR_REB          1
   START_STATISTIC  1
   START_TIMER      1
END

# vlastní experiment
BEGIN
   OPERATIONS       100000    #*
   INSERT           60
   DELETE           30
   FIND             10
   REB_PERIOD       500       #*
END

# ukončení měření času
BEGIN
   STOP_TIMER       1
END
```

Hodnoty proměnných na řádcích obsahující v komentáři znak * se mění v závislosti na počtu zadaných operací nebo na různých vyvažovacích periodách.

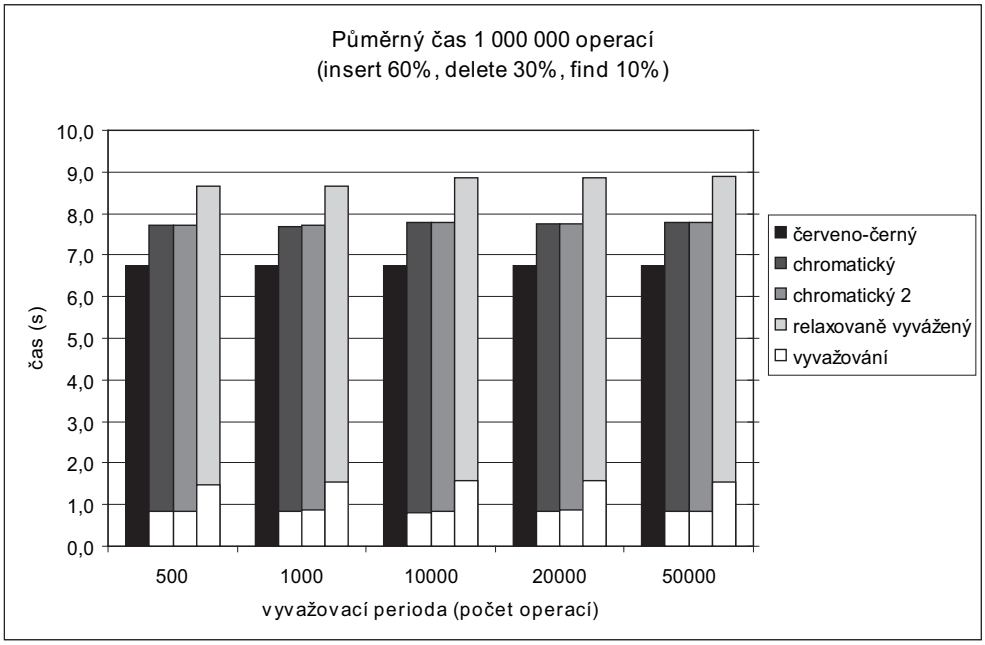
Poznámka. Ostatní testy, které jsou prováděny na předem inicializovaném vyváženém červeno-černém stromu, používají stejné skripty kromě úpravy bloku s vlastním experimentem (respektive s jinou hodnotou velikosti úvodního stromu, např. experiment v kapitole 5.5).



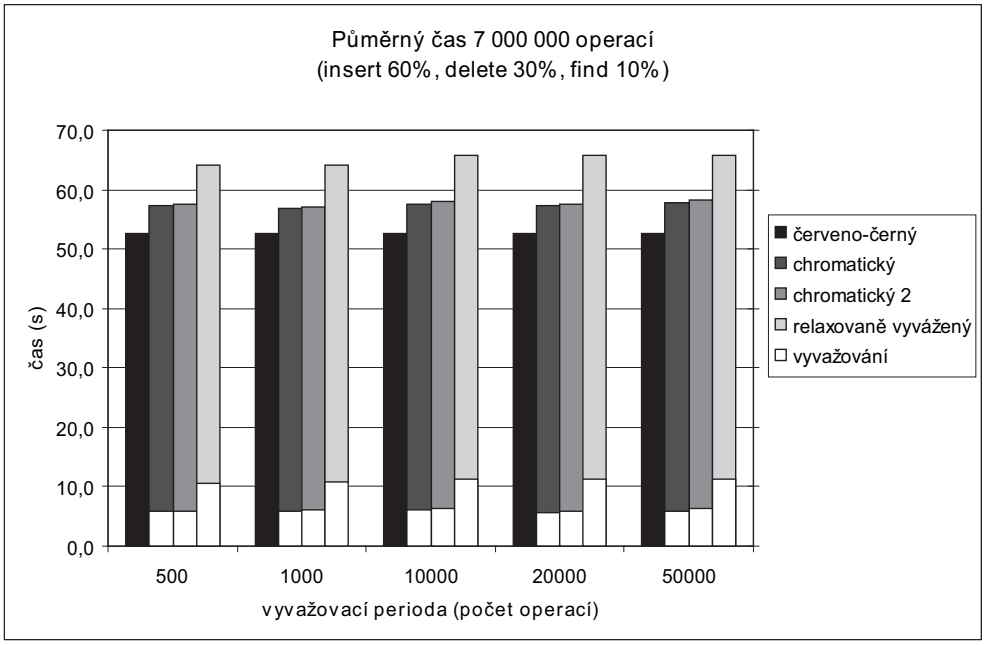
Graf 1

Výsledky experimentů nad počtem operací 100000, 1000000 a 7000000 zobrazují grafy 1 – 3. Pokud se zaměříme na srovnání striktního červeno-černého stromu s relaxovanými verzemi, zjistíme, že ve všech uvedených případech vyznívá časová náročnost jednoznačně ve prospěch RB stromu. Provedené testy tedy skončí dříve než při použití relaxavaného vyvažování.

Dále se soustřeďme na rozdíly mezi jednotlivými druhy relaxovaných červeno-černých stromů. Obecně lze z výsledků vypozorovat, že se zvyšující vyvažovací periodou mírně roste doba potřebná k výpočtu testů, a to bez


Půměrný čas 1 000 000 operací
(insert 60%, delete 30%, find 10%)
10,0
9,0
8,0
7,0
6,0
5,0
4,0
3,0
2,0
1,0
0,0
čas (s)
500
1000
10000
20000
50000
vyvažovací perioda (počet operací)
červeno-černý
chromatický
chromatický 2
relaxovaně vyvážený
vyvažování

Graf 2


Půměrný čas 7 000 000 operací
(insert 60%, delete 30%, find 10%)
70,0
60,0
50,0
40,0
30,0
20,0
10,0
0,0
čas (s)
500
1000
10000
20000
50000
vyvažovací perioda (počet operací)
červeno-černý
chromatický
chromatický 2
relaxovaně vyvážený
vyvažování

Graf 3

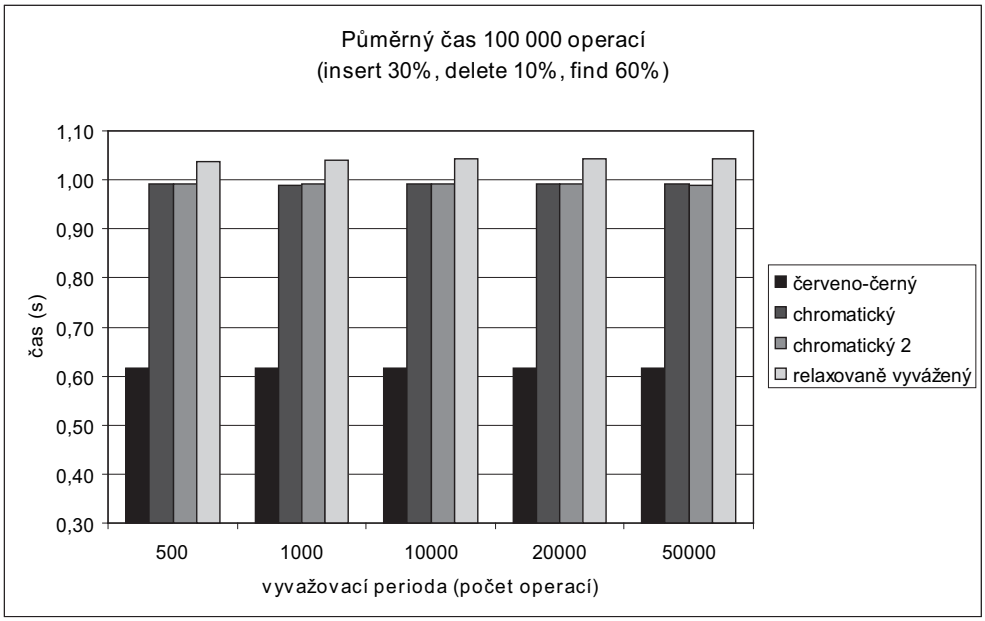
rozdílu na typu stromu. Samotná časová náročnost vychází nejlépe pro oba typy chromatického stromu, které jsou výrazně rychlejší než zbývající relaxovaně vyvážený strom. Odlišnosti mezi oběma chromatickými stromy jsou pouze nepatrné.

Na shodných vstupních datech jsme dále měřili celkový čas, který daná struktura spotřebuje na samotné vyvažovažování do podoby červeno-černého stromu. Výsledky získaných hodnot jsou pro lepší názornost zachyceny ve stejných grafech jako předchozí experiment. Největší čas zabírá vyvažovací operace u relaxovaně vyváženého stromu. Oba typy chromatického stromu vykazují téměř totožné časy. Menší periody dosahují zanedbatelně nižších hodnot než v případě vyvažování po větších intervalech.

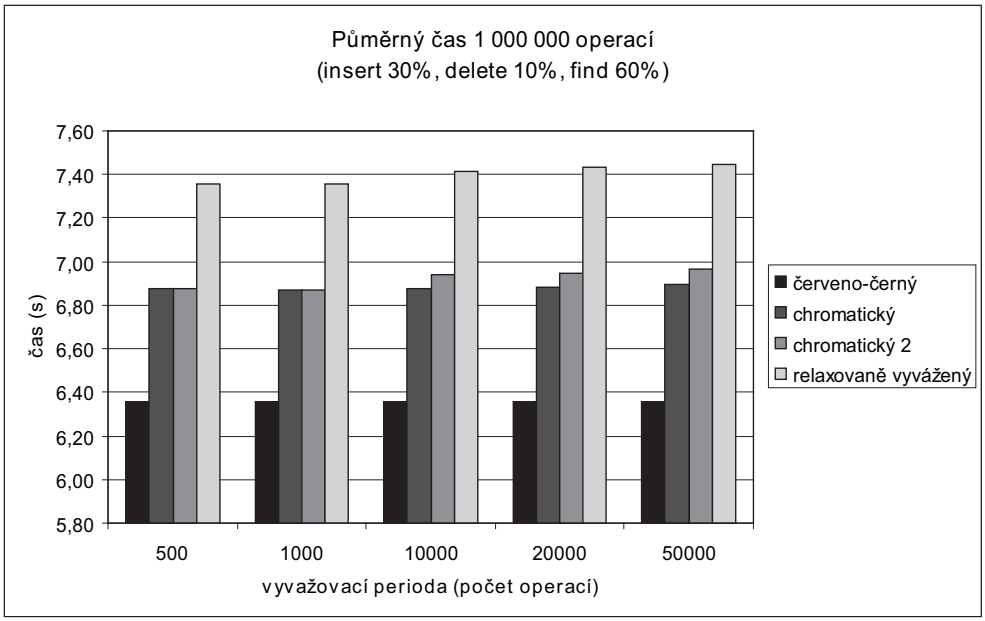
5.3 Obecný test 2

Tento experiment se podobá předchozímu. Rozdíl je ovšem v procentuálním zastoupení jednotlivých operací aplikovaných na popisované struktury. Zaměřujeme se spíše na měření času v prostředí, ve kterém převládá vyhledávání dat. Tudíž jsme zvolili následující rozložení pro generování prováděných operací: insert 30%, delete 10% a find 60%.

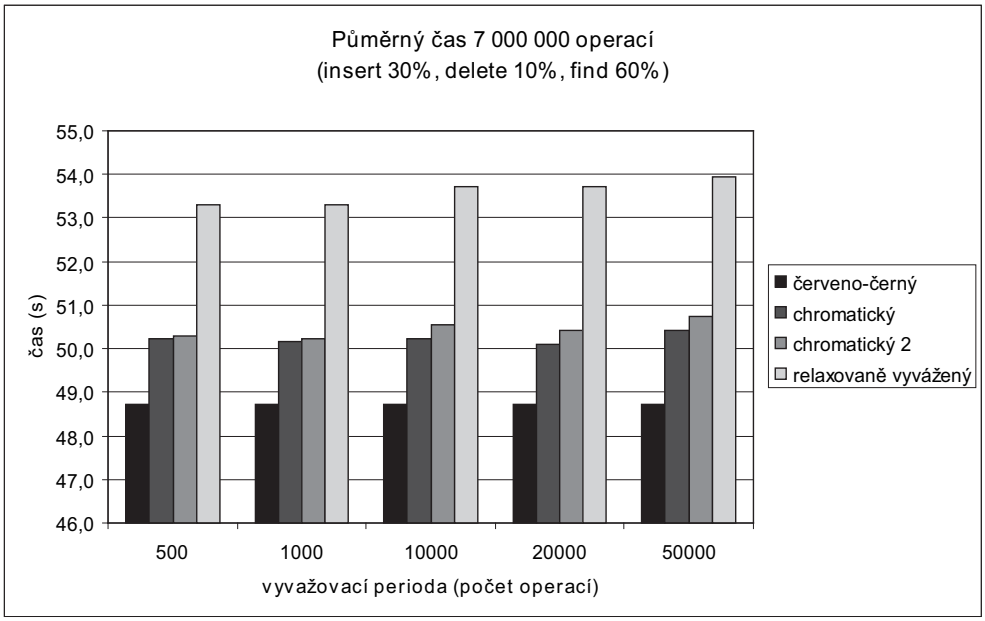
Porovnání naměřených výsledků (grafy 4 – 6) opět vyznívá jednoznačně nejlépe ve prospěch standardního červeno-černého stromu. Mezi relaxovanými stromy experimenty skončí dříve u obou chromatických verzí než u třetí struktury. Nepatrně lepších časů dosahuje chromatický strom typu 1 oproti typu 2. Jejich rozdíl je ovšem ve většině testů minimální. Dále vidíme, že nižší vyvažovací periody mají opět lepší výsledky než vyvažování po více operacích. Tento rozdíl je znatelný na experimentech s větším objemem dat.



Graf 4


Půměrný čas 1 000 000 operací
(insert 30%, delete 10%, find 60%)
7,60
7,40
7,20
7,00
6,80
6,60
6,40
6,20
6,00
5,80
čas (s)
500
1000
10000
20000
50000
vyvažovací perioda (počet operací)
červeno-černý
chromatický
chromatický 2
relaxovaně vyvážený

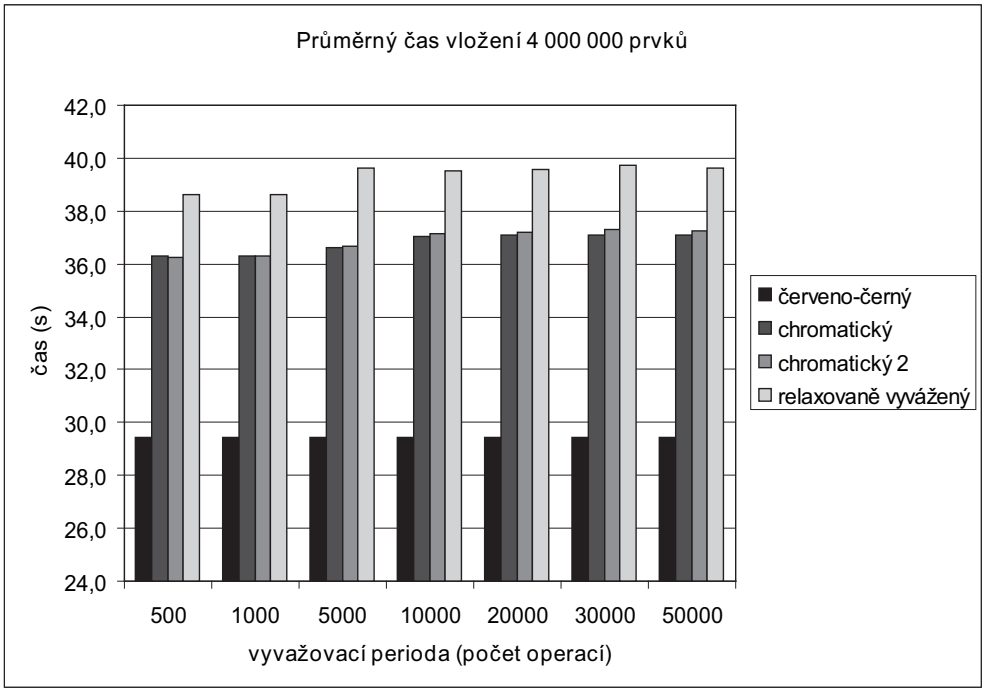
Graf 5


Půměrný čas 7 000 000 operací
(insert 30%, delete 10%, find 60%)
55,0
54,0
53,0
52,0
51,0
50,0
49,0
48,0
47,0
46,0
čas (s)
500
1000
10000
20000
50000
vyvažovací perioda (počet operací)
červeno-černý
chromatický
chromatický 2
relaxovaně vyvážený

Graf 6

5.4 Měření průměrného času operace insert

Dalším testem je vkládání 4000000 prvků do předem inicializovaného červeno-černého stromu o velikosti 1000000. Vyvažování jsme prováděli vždy pravidelně po 500, 1000, 5000, 10000, 20000, 30000 nebo 50000 krocích. Cílem experimentu bylo porovnat operace *insert* z hlediska celkové časové náročnosti. Shrnutí naměřených hodnot pro porovnání všech RB stromů zobrazuje graf 7. Opět z něj vyplývá jasná výhoda standardního červeno-černého stromu oproti ostatním strukturám. Oba chromatické stromy, kde typ 1 je v některých výsledcích zanedbatelně výhodnější, vykazují podstatně rychlejší provádění testů oproti relaxovaně vyváženému RB stromu.

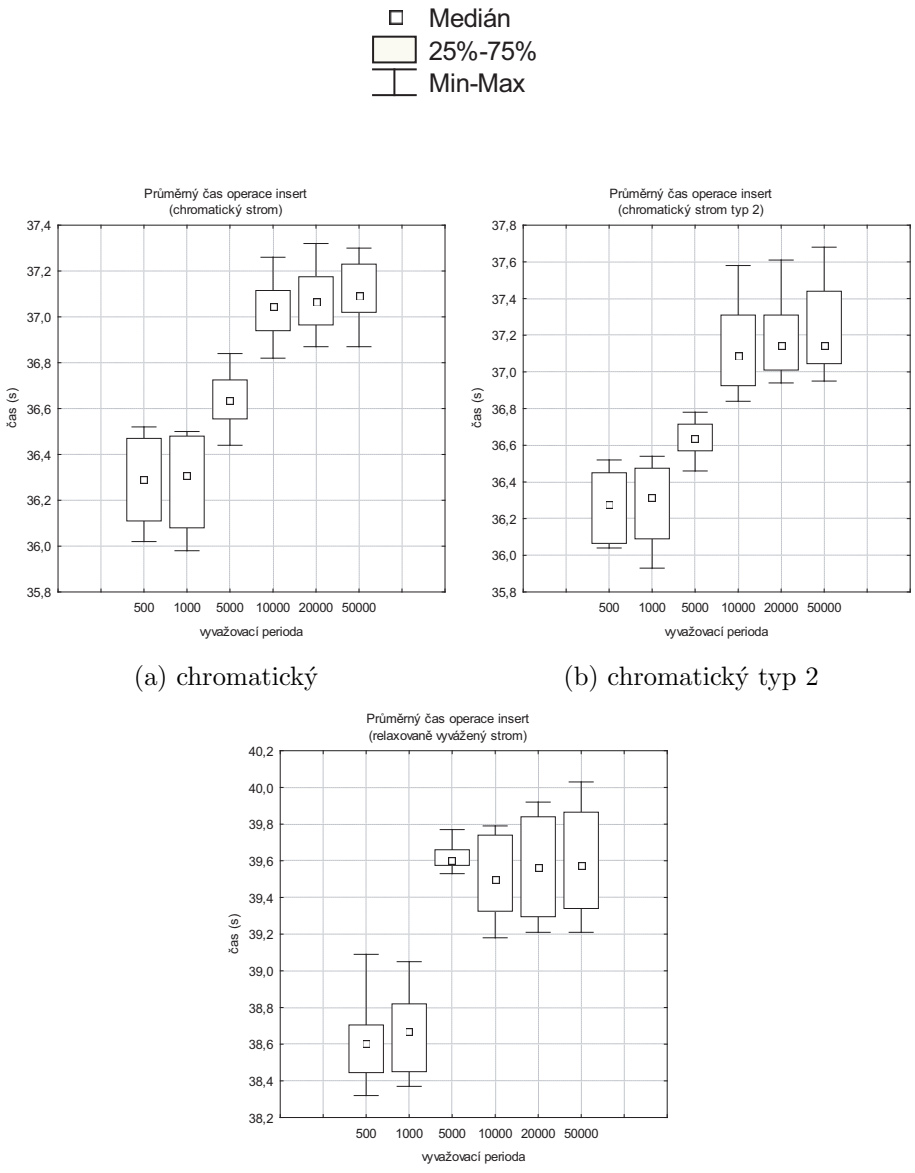


Graf 7

Na tzv. krabicových grafech závislosti celkového spotřebovaného času na vyvažovací periodě (viz graf 8) lze vidět rozložení výsledků pro jednotlivé struktury relaxovaných RB stromů. Snadno z nich můžeme vypozorovat nárůst časové náročnosti se zvětšující se periodou vyvažování. Ovšem v tomto

případě je rozdíl nejmenší periody (500 operací) proti největší (50000) znatelný.

Popis všech krabicových grafů v této práci, které zachycují statistické rozdělení naměřených hodnot, odpovídá následujícímu obrázku:



(a) chromatický

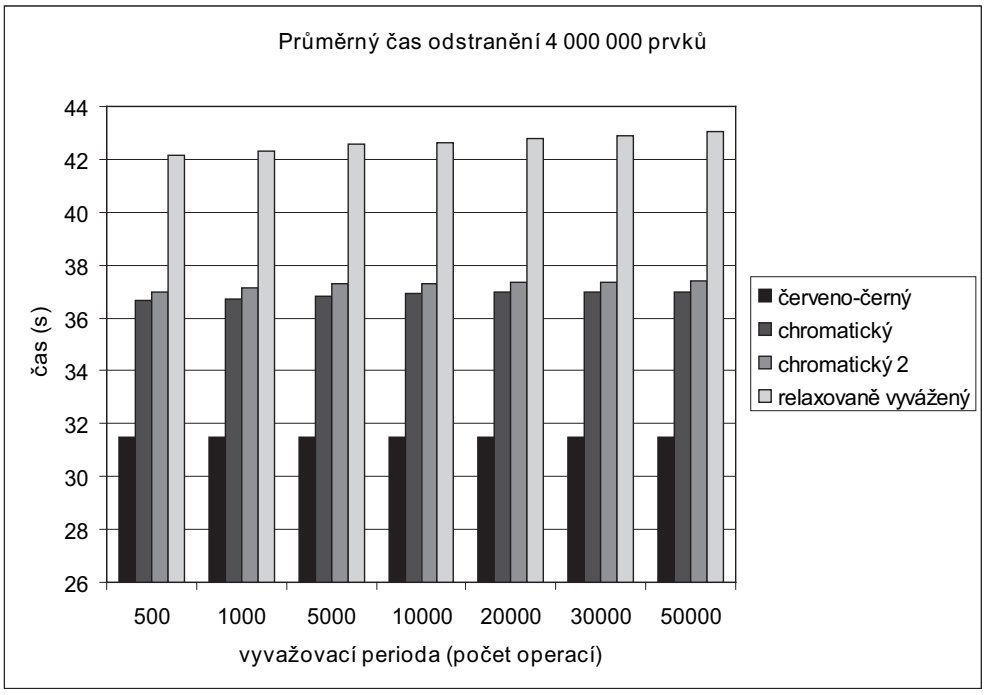
(b) chromatický typ 2

(c) relaxovaně vyvážený

Graf 8

5.5 Měření průměrného času operace delete

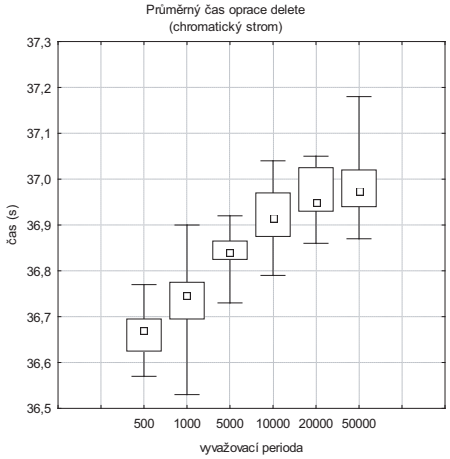
Následující experiment je obdobný předchozímu, ale s tím rozdílem, že měříme čas 4000000 operací *delete* prováděných na počátečním RB stromu o velikosti 5000000 prvků. Cílem tedy bylo otestovat chování popisovaných datových struktur s ohledem na časovou náročnost mazání prvků. Výsledky jsou téměř totožné – relaxovaně vyvážený strom zaostává oproti chromatickým strukturám, ale červeno-černý strom jasně všechny tyto stromy předčí. Rozdíl můžeme zaznamenat pouze v rychlejším provedení testu v případě chromatickeho stromu typu 1 proti typu 2. Také nižší perioda vyvažování nemá tak znatelný vliv na dřívější dokončení pokusu jako v situaci vkládání prvků z předchozího odstavce. Výjimku v tomto směru tvoří relaxovaně vyvážený RB strom. Porovnání všech stromů zachycuje graf 9.



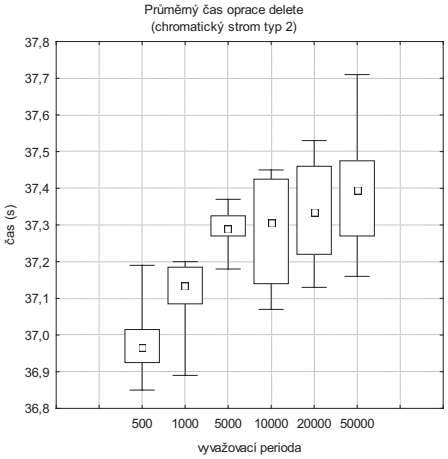
Graf 9

Detailní pohled pro jednotlivé struktury znázoňují grafy na obrázku 10. V případě relaxovaně vyváženého stromu může pozorovat velice pomalé pro-

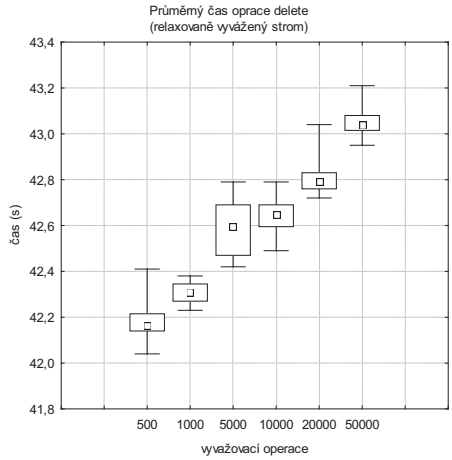
vádění operací *delete* oproti vkládání prvků, které je u zbývajících námi testovaných stromů téměř srovnatelně rychlé s tímto experimentem.



(a) chromatický



(b) chromatický typ 2

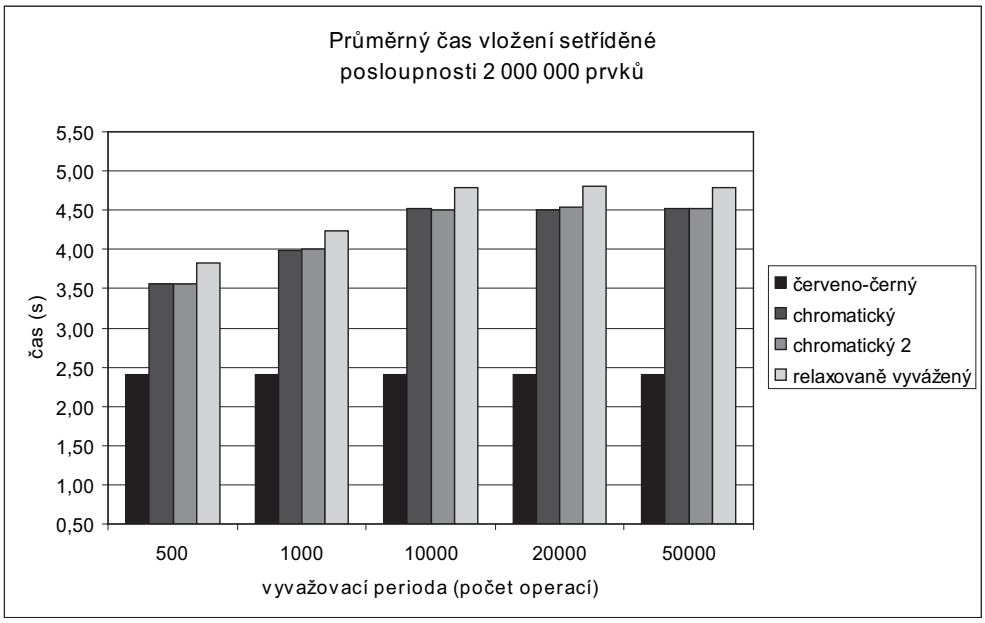


(c) relaxovaně vyvážený

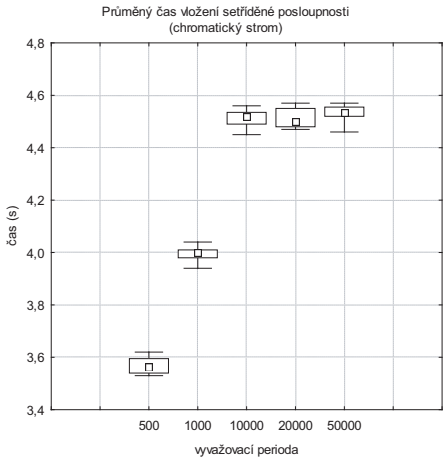
Graf 10

5.6 Měření průměrného času vložení setříděné posloupnosti

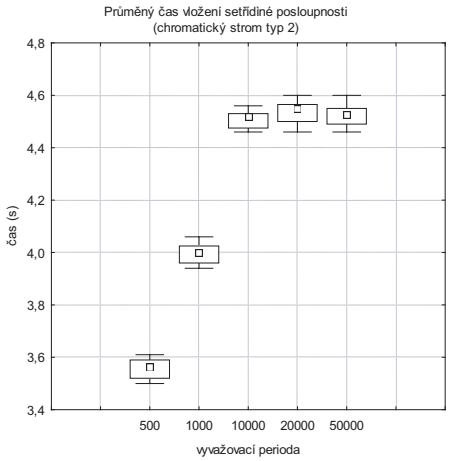
Posledním provedeným testem bylo měření průměrného času vložení setříděné posloupnosti 2000000 hodnot do počátečního stromu obsahující 1000000 prvků. Výsledné grafy 11 a 12 poskytují obdobný obrázek jako všechny dříve zmíněných experimenty. Červeno-černý strom vykoná požadované operace nejrychleji, oba chromatické stromy mají téměř shodné časy a relaxovaně vyvážený RB strom opět zůstává poslední. Může zde ovšem nají několik odlišností. Nižší perioda vyvažování má u všech struktur značně lepší efekt na konečný výsledek než vyšší a rozdíl mezi relaxovanými strukturami není tak znatelný, jako v jiných experimentech.



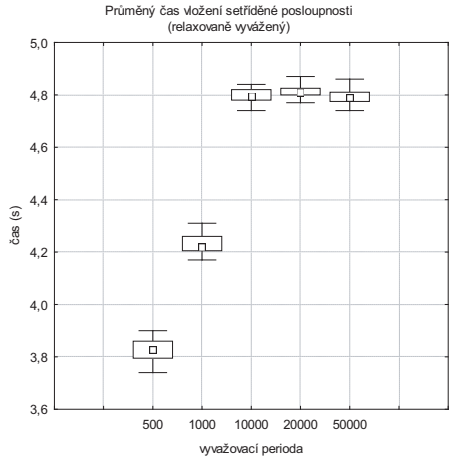
Graf 11



(a) chromatický



(b) chromatický typ 2

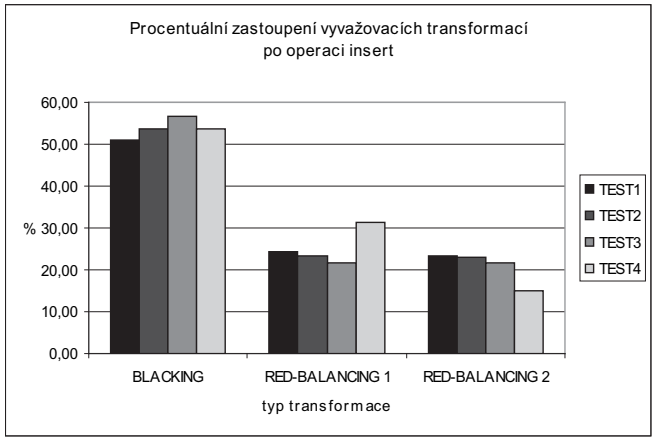


(c) relaxovaně vyvážený

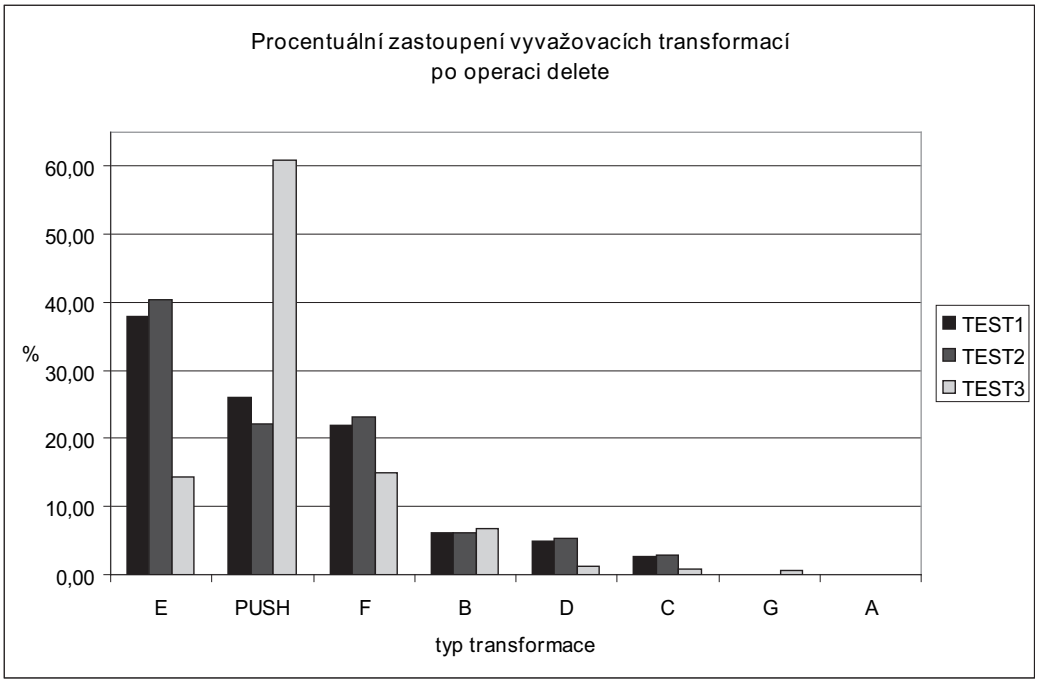
Graf 12

5.7 Procentuální zastoupení jednotlivých typů transformací

Na závěr všech experimentů si uvedeme porovnání jednotlivých vyvažovacích transformací prováděných při testech z hlediska četnosti zastoupení mezi celkovým počtem vykonaných úprav. Výsledky zachycují grafy 13 a 14. Na prvním z nich můžeme vidět vyvažovací operace volané po vložení nové hodnoty do stromu a na druhém vyvažovací úpravy v důsledku odstranění prvku. Nárůst počtu operací *PUSH* v testu číslo 3 při mazání prvků je způsoben tím, že tento experiment produkuje velké množství situací, které řeší právě zmíněná transformace.



Graf 13: blacking (obr. 4.22), red-balancing 1 (obr. 4.23a), red-balancing 2 (obr. 4.23b)



Graf 14: Označení transformací v grafu odpovídá nákresům pod stejnými písmeny na obrázku 4.26. Kromě úpravy *PUSH*, kterou můžeme nalézt na obrázku 4.25.

6 Závěr

Cílem této diplomové práce bylo porovnat standardní červeno-černé stromy s jejich relaxovanými verzemi, které byly primárně sestaveny pro využití v paralelním prostředí, kde svoji výhodu, jak bylo uvedeno v [5] a prezentováno v kapitole 4.7, prakticky dokázaly. My jsme se ale v experimentech zaměřili na chování při neparalelním běhu. Výsledky všech našich testů ovšem dopadly jednoznačně nejlépe ve prospěch standardní varianty. Její výhoda oproti relaxovaným strukturám je na první pohled zřejmá a můžeme říci, že docela výrazná. Jako efektivnější se tedy ukázalo provést vyvážení ihned po aktualizaci ve stromě.

Další otázkou, na níž nám měla naše studie odpovědět, bylo, který relaxovaný typ červeno-černého stromu bude vykazovat v testech nejpřijatelnější hodnoty, tj. nejmenší naměřené časy. V tomto směru vyzněl nejhůře relaxovaně vyvážený strom. Z uvedených grafů lze snadno vyčíst, že rozdíl je opravdu značný, i když množina jeho transformací obsahuje méně operací než u zbývajících dvou struktur. Z nich vyšel jako vítěz chromatický strom typu 1, ale pouze se zanedbatelnou výhodou oproti variantě číslo 2. V některých experimentech byly jejich výsledky téměř totožné.

Pokud se zaměříme na hodnocení výhodnosti nižší nebo vyšší vyvažovací periody, dojdeme k jednoznačnému výsledku. Ve všech testech dopadla měření lépe pro konfiguraci s vyvažováním po menším počtu operací a v některých experimentech byl tento rozdíl opravdu výrazný.

V první polovině se nám také podařilo shrnout informace o červeno-černých stromech a o dosud prezentovaných relaxovaných verzích a implementovat tyto poznatky v programu, který posloužil jako nástroj pro experimentální měření daných algoritmů. Dále jsme vytvořili jednoduchý skriptovací jazyk, pomocí něhož je možné snadno definovat potřebný test nebo vygenerovat posloupnost operací pro pozdější provedení experimentu.

Literatura

[1] J. Boyar, R. Fagerberg a K. S. Larsen. Amortization results for chromatic search trees, with an application to priotiry queues. *Journal of Computer and System Sciences*, 55:504–521, 1997.

[2] J. Boyar a K. S. Larsen. Efficient rebalancing of chromatic search trees. *Journal of Computer and System Sciences*, 49(3):667–682, 1993.

[3] L. J. Guibas a R. Sedgewick. A dichromatic framework for balanced trees. *FOES*, 19:8–21, 1978.

[4] S. Hanke. Chromatic search trees revisited. Technical Report report00090, Institut für Informatik, Universitat Freiburg, Germany, 12, 1997.

[5] S. Hanke. The performance of concurrent red-black tree algorithms. *Lecture Notes in Computer Science*, 1668:286–300, 1999.

[6] S. Hanke, T. Ottmann a E. Soisalon-Soininen. Relaxed balanced red-black trees. *CIAC*, pages 193–204, 1997.

[7] S. Hanke a E. Soisalon-Soininen. Group updates for red-black trees. *Acta Informatica*, 38(8):565–586, 2002.

[8] K. S. Larsen. Amortized constant relaxed reblancing using standard rotation. *Acta Informatica*, 35(10):859–874, 1998.

[9] K. S. Larsen. Relaxed red-black trees with group updates. *Acta Informatica*, 38:565–586, 2002.

[10] K. S. Larsen, T. Ottmann a E. Soisalon-Soininen. Relaxed balance for search trees with local rebalancing. *Acta Informatica*, 37(10):743–763, 2001.

[11] L. Malmi. On updating and rebalancing relaxed balanced search trees in main memory. Technical Report TKO-A-35, Helsinki University of Technology, 1997.

[12] O. Nurmi a E. Soisalon-Soininen. Uncoupling updating and rebalancing in chromatic binary search trees. *ACM Press*, pages 192–198, 1991.

[13] O. Nurmi a E. Soisalon-Soininen. Chromatic binary search trees. *Acta Informatica*, 33:547–557, 1996.

[14] O. Nurmi, E. Soisalon-Soininen a D. Wood. Concurrency control in database structures with relaxed balance. *PODS '87: Proceedings of the sixth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 170–176, 1987.

[15] T. Ottmann a E. Soisalon-Soininen. Relaxed balancing made simple. Technical Report report00071, Albert-Ludwigs University at Freiburg, 1, 1995.

A Tabulky

Tato příloha obsahuje části výsledků z provedených experimentů. Zbývající lze nalézt na přiloženém CD.

Popis uvedených názvů:

průměr průměrný čas vypočítaný z naměřených hodnot
min nejnižší naměřená hodnota
max nejvyšší naměřený čas
dev směrodatná odchylka

Tab. 1: Průměrný čas 100 000 operací
(insert 60%, delete 30%, find 10%)

perioda		chromatický	chromatický 2	relaxovaný	červeno-černý
500	průměr	1,080500	1,081500	1,178500	0,646500
	min	1,040000	1,040000	1,150000	0,630000
	max	1,120000	1,120000	1,200000	0,660000
	dev	0,029820	0,030826	0,020590	0,008751
1000	průměr	1,082500	1,078000	1,180000	0,646500
	min	1,040000	1,040000	1,150000	0,630000
	max	1,120000	1,110000	1,200000	0,660000
	dev	0,031267	0,028210	0,017472	0,008751
10000	průměr	1,086500	1,084000	1,194000	0,646500
	min	1,050000	1,040000	1,160000	0,630000
	max	1,120000	1,130000	1,220000	0,660000
	dev	0,033604	0,032671	0,021619	0,008751
20000	průměr	1,086000	1,085500	1,197000	0,646500
	min	1,040000	1,050000	1,170000	0,630000
	max	1,120000	1,130000	1,230000	0,660000
	dev	0,031523	0,031031	0,022501	0,008751
50000	průměr	1,083500	1,075500	1,183000	0,646500
	min	1,040000	1,040000	1,150000	0,630000
	max	1,120000	1,110000	1,210000	0,660000
	dev	0,033760	0,029820	0,020800	0,008751

Tab. 2: Průměrný čas 1 000 000 operací
(insert 60%, delete 30%, find 10%)

perioda		chromatický	chromatický 2	relaxovaný	červeno-černý
500	průměr	7,705000	7,725000	8,663000	6,733500
	min	7,630000	7,630000	8,580000	6,700000
	max	7,780000	7,830000	8,730000	6,790000
	dev	0,063287	0,068939	0,043054	0,025808
1000	průměr	7,696500	7,711000	8,667000	6,733500
	min	7,590000	7,620000	8,610000	6,700000
	max	7,770000	7,790000	8,740000	6,790000
	dev	0,055751	0,067582	0,041052	0,025808
10000	průměr	7,771000	7,792500	8,875500	6,733500
	min	7,670000	7,690000	8,810000	6,700000
	max	7,870000	7,880000	8,940000	6,790000
	dev	0,079532	0,069348	0,032843	0,025808
20000	průměr	7,743000	7,765500	8,872500	6,733500
	min	7,660000	7,660000	8,820000	6,700000
	max	7,840000	7,830000	8,920000	6,790000
	dev	0,068295	0,060912	0,021975	0,025808
50000	průměr	7,771000	7,793000	8,881000	6,733500
	min	7,660000	7,710000	8,830000	6,700000
	max	7,880000	7,870000	8,920000	6,790000
	dev	0,072104	0,061224	0,029001	0,025808

Tab. 3: Průměrný čas 7 000 000 operací

(insert 60%, delete 30%, find 10%)

perioda		chromatický	chromatický 2	relaxovaný	červeno-černý
500	průměr	57,362500	57,468000	64,069000	52,539000
	min	56,960000	56,980000	63,860000	52,110000
	max	57,790000	57,940000	64,320000	52,870000
	dev	0,306609	0,347496	0,147894	0,253894
1000	průměr	56,960500	57,108500	64,148000	52,539000
	min	56,840000	56,940000	63,970000	52,110000
	max	57,170000	57,240000	64,310000	52,870000
	dev	0,087146	0,087074	0,107928	0,253894
10000	průměr	57,517500	57,909000	65,709500	52,539000
	min	57,330000	57,520000	65,460000	52,110000
	max	57,700000	58,400000	65,990000	52,870000
	dev	0,100414	0,295793	0,143581	0,253894
20000	průměr	57,259500	57,623000	65,708000	52,539000
	min	56,660000	56,990000	65,190000	52,110000
	max	57,700000	58,050000	66,540000	52,870000
	dev	0,352696	0,364693	0,385208	0,253894
50000	průměr	57,866500	58,180500	65,836000	52,539000
	min	57,420000	57,640000	65,420000	52,110000
	max	58,370000	58,490000	66,300000	52,870000
	dev	0,389443	0,315269	0,282813	0,253894

Tab. 4: Průměrný čas 100 000 operací
(insert 30%, delete 10%, find 60%)

perioda		chromatický	chromatický 2	relaxovaný	červeno-černý
500	průměr	0,993000	0,992000	1,038500	0,616000
	min	0,970000	0,970000	1,030000	0,610000
	max	1,010000	1,010000	1,050000	0,630000
	dev	0,013416	0,011517	0,007452	0,006806
10000	průměr	0,991000	0,992500	1,044000	0,616000
	min	0,970000	0,970000	1,030000	0,610000
	max	1,010000	1,010000	1,050000	0,630000
	dev	0,012937	0,013717	0,006806	0,006806
50000	průměr	0,993000	0,989000	1,043500	0,616000
	min	0,970000	0,970000	1,030000	0,610000
	max	1,010000	1,010000	1,060000	0,630000
	dev	0,014179	0,014105	0,007452	0,006806

Tab. 5: Průměrný čas 1 000 000 operací
(insert 30%, delete 10%, find 60%)

perioda		chromatický	chromatický 2	relaxovaný	červeno-černý
500	průměr	6,873000	6,879000	7,357500	6,355000
	min	6,850000	6,850000	7,330000	6,300000
	max	6,890000	6,900000	7,390000	6,410000
	dev	0,009787	0,015183	0,016819	0,034105
10000	průměr	6,878000	6,940500	7,415000	6,355000
	min	6,850000	6,880000	7,390000	6,300000
	max	6,890000	7,010000	7,440000	6,410000
	dev	0,012814	0,056798	0,014690	0,034105
50000	průměr	6,892500	6,964500	7,447000	6,355000
	min	6,870000	6,880000	7,420000	6,300000
	max	6,910000	7,040000	7,470000	6,410000
	dev	0,011642	0,063202	0,014903	0,034105

Tab. 6: Průměrný čas 7 000 000 operací
(insert 30%, delete 10%, find 60%)

perioda		chromatický	chromatický 2	relaxovaný	červeno-černý
500	průměr	50,233500	50,307000	53,300500	48,718500
	min	50,140000	50,180000	53,180000	48,460000
	max	50,340000	50,470000	53,420000	49,040000
	dev	0,058963	0,078814	0,062448	0,182822
10000	průměr	50,230500	50,543500	53,727000	48,718500
	min	50,130000	50,210000	53,570000	48,460000
	max	50,360000	50,870000	53,950000	49,040000
	dev	0,062616	0,254419	0,086457	0,182822
50000	průměr	50,420500	50,755000	53,940000	48,718500
	min	50,280000	50,440000	53,850000	48,460000
	max	50,550000	51,100000	54,020000	49,040000
	dev	0,063700	0,282200	0,050680	0,182822

Tab. 7: Průměrný čas vložení 4 000 000 prvků

perioda		chromatický	chromatický 2	relaxovaný	červeno-černý
500	průměr	36,283000	36,272500	38,605500	29,439000
	min	36,020000	36,040000	38,320000	29,260000
	max	36,520000	36,520000	39,090000	29,580000
	dev	0,189239	0,194527	0,189444	0,101717
10000	průměr	37,030500	37,129000	39,523500	29,439000
	min	36,820000	36,840000	39,180000	29,260000
	max	37,260000	37,580000	39,790000	29,580000
	dev	0,119185	0,244344	0,222410	0,101717
50000	průměr	37,110500	37,229500	39,600000	29,439000
	min	36,870000	36,950000	39,210000	29,260000
	max	37,300000	37,680000	40,030000	29,580000
	dev	0,141104	0,247226	0,283549	0,101717

Tab. 8: Průměrný čas odstranění 4 000 000 prvků

perioda		chromatický	chromatický 2	relaxovaný	červeno-černý
500	průměr	36,666500	36,973500	42,178500	31,490000
	min	36,570000	36,850000	42,040000	31,350000
	max	36,770000	37,190000	42,410000	31,620000
	dev	0,054219	0,076865	0,078289	0,090029
10000	průměr	36,920000	37,285500	42,641500	31,490000
	min	36,790000	37,070000	42,490000	31,350000
	max	37,040000	37,450000	42,790000	31,620000
	dev	0,060350	0,138278	0,077410	0,090029
50000	průměr	36,986000	37,378500	43,049000	31,490000
	min	36,870000	37,160000	42,950000	31,350000
	max	37,180000	37,710000	43,210000	31,620000
	dev	0,075770	0,139181	0,061550	0,090029

Tab. 9: Průměrný čas vložení setříděné posloupnosti 2000000 prvků

perioda		chromatický	chromatický 2	relaxovaný	červeno-černý
500	průměr	3,568000	3,557500	3,824500	2,410500
	min	3,530000	3,500000	3,740000	2,360000
	max	3,620000	3,610000	3,900000	2,450000
	dev	0,030366	0,037819	0,041735	0,031867
10000	průměr	4,512500	4,510000	4,797000	2,410500
	min	4,450000	4,460000	4,740000	2,360000
	max	4,560000	4,560000	4,840000	2,450000
	dev	0,033226	0,033717	0,030105	0,031867
50000	průměr	4,528000	4,521500	4,792000	2,410500
	min	4,460000	4,460000	4,740000	2,360000
	max	4,570000	4,600000	4,860000	2,450000
	dev	0,034428	0,036023	0,030192	0,031867

B Struktura přiloženého CD

Na přiloženém CD se nachází tyto hlavní adresáře:

`config`	konfigurační soubory jednotlivých experimentů
`doc`	text diplomové práce a vygenerovaná dokumentace k programu
`excel`	grafy a tabulky výsledků experimentů zpracované programem *Microsoft Excel*
`images`	obrázky použité v diplomové práci ve formátu eps
`results`	výstupy z měření
`source`	zdrojový kód programu a pomocné skripty na zpracování výsledků
`statistica`	grafy a tabulky výsledků experimentů zpracované programem *STATISTICA*
`test`	soubor s ukázkovými daty jednoho z experimentů

Jednotlivé adresáře mohou obsahovat podadresáře a soubory `Readme.txt` s popisem jejich obsahu.

C Kompilace a spuštění programu

V této části přílohy si popíšeme kompilaci a spouštění programu včetně rozboru všech jeho parametrů. Zdrojové kódy jsou k dispozici na přiloženém CD.

Přeložení zdrojových kódů probíhá pomocí kompilátoru `g++`. Pro použití utility `make` nalezneme na CD v adresáři `source` již vytvořený soubor `Makefile`, ve kterém jsou popsány všechny potřebné závislosti jednotlivých souborů. Výsledný spustitelný program se vytváří v adresáři `/source/bin`.

Program má pět variant překladu, které jsou ovlivněny přepínači zapnutými při kompilaci a mají vliv na chování samotné aplikace. Pro všechny z nich lze aplikaci spouštět s následujícími parametry:

- `-c file` vstupní konfigurační soubor pro generování dat
- `-f file` vstupní soubor pro experiment obsahující již vygenerovaná data (nelze kombinovat společně s `-c`, `-g` a `-o`)
- `-h` zobrazení nápovědy
- `-o file` definice výstupního souboru při generování testovacích dat (nelze použít bez parametru `-c`)
- `-r file` výstupní soubor pro uložení výsledku provedených testů
- `-t tree` typ stromu, na kterém bude experiment probíhat
  - `CHR` – chromatický
  - `CHR2` – chromatický typ 2
  - `REL` – relaxovaný
  - `RB` – červeno-černý

Další parametry jsou přístupné pouze po překladu s přepínačem `-DLOG` a slouží k zapnutí vypisování ladících a informačních zpráv při vykonávání testu (přesný popis je k dispozici na CD). V našem souboru `Makefile` jsme pro tuto volbu definovali pravidlo `log`, tudíž příkaz `make log` zajistí jejich funkčnost.

Dalšími módy překladu jsou varianta s přepínačem `-DTEST`, která zkompiluje program počítající provedené operace (v souboru `Makefile` pravidlo `test`), a `-DT_REB` (pravidlo `rebalance`) zapínající sčítání času stráve-

ného vyvažováním. A konečně kompilace bez přepínačů (pravidlo `timer`) slouží pro vytvoření binárního souboru programu umožňující testování celkového spotřebovaného času bez vedlejších vlivů, jako je logování, počítání prováděných operací atd.

Implicitní volbou při kompilaci bez zadaného cíle (tj. pouze příkazem `make` v adresáři `source`) je překlad se zapnutými všemi přepínači.

Když nebereme v úvahu různé kombinace vypisování informačních zpráv, existují tři druhy volání programu:

- provedení experimentu na datech z předem připraveného souboru
  ```
  ./main -f file -t tree
  ```
- provedení testu na datech vygenerovaných před samotným provedením experimentu podle konfiguračního souboru
  ```
  ./main -c config -t tree
  ```
- pouhé vygenerování dat pro experiment na základě zadaného konfiguračního souboru a jejich uložení
  ```
  ./main -g -c config -o file
  ```

U všech variant lze ještě navíc použít parametr `-r`.