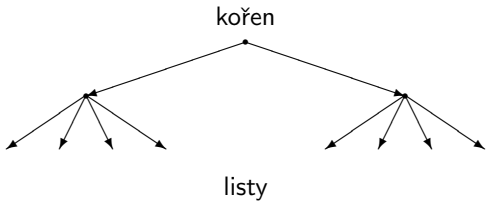
Binární vyhledávací strom a jeho varianty

Michal Žemlička

1 Stromy

V přednáškách z teorie grafů byl strom popsán mj. jako souvislý graf bez cyklů. Zde se budeme zabývat orientovanými zakořeněnými stromy – takovými, kde hrany jsou orientovány směrem od kořene k listům (viz obr. 1).



Obrázek 1: Příklad orientovaného stromu

Mějme hranu vedoucí z uzlu u do uzlu v. Uzel u pak můžeme nazvat předkem v, uzel v pak *potomkem* uzlu u.

Je-li možné omezit pro celý strom počet přímých potomků každého z uzlů číslem k, označujeme takový strom jako k-ární. Zde se soustředíme zejména na stromy binární – tedy s takové, kde uzly mají nejvýše dva potomky.

Obrázek 2: Příklad binárního stromu

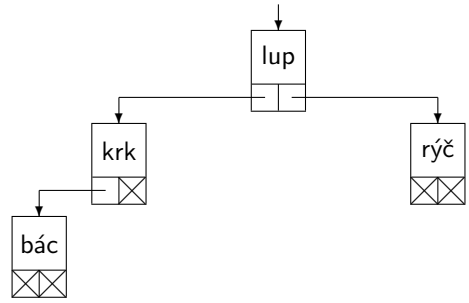
Uzly je možné doplnit o hodnotu – *ohodnotit*. Ohodnoceným stromem můžeme reprezentovat *kolekci* (kdy se prvky mohou opakovat) nebo množinu, kdy je každé ohodnocení ve stromě nejvýše jednou. Nebude-li řečeno jinak, budeme v příkladech uvažovat reprezentaci množiny.

1.1 Binární vyhledávací strom

Mají-li být jednotlivá ohodnocení snadno ve stromě nalezitelná, je vhodné, aby byla uspořádaná. Platí-li pro každý uzel v binárním stromě, že ohodnocení potomků v jeho levém podstromě jsou menší, než ohodnocení jeho samého, a ohodnocení v pravém podstromě jsou větší, pak takový strom nazveme *binárním vyhledávacím stromem*.

Na binárním vyhledávacím stromě jsou definovány tyto operace:

- vytvoření prázdného stromu
- zrušení
- přidání prvku



Obrázek 3: Příklad binárního vyhledávacího stromu

- vyřazení prvku
- je strom prázdný?
- je ve stromě daný prvek?
- nalezení prvku
- průchod stromem dle pořadí hodnot (inorder)
- průchod stromem „nejdříve uzel, pak jeho potomci“ (preorder)
- průchod stromem „nejdříve potomci, pak uzel“ (postorder)

Operace na stromě jsou většinou definovány rekurentně a tak bývají i implementovány.

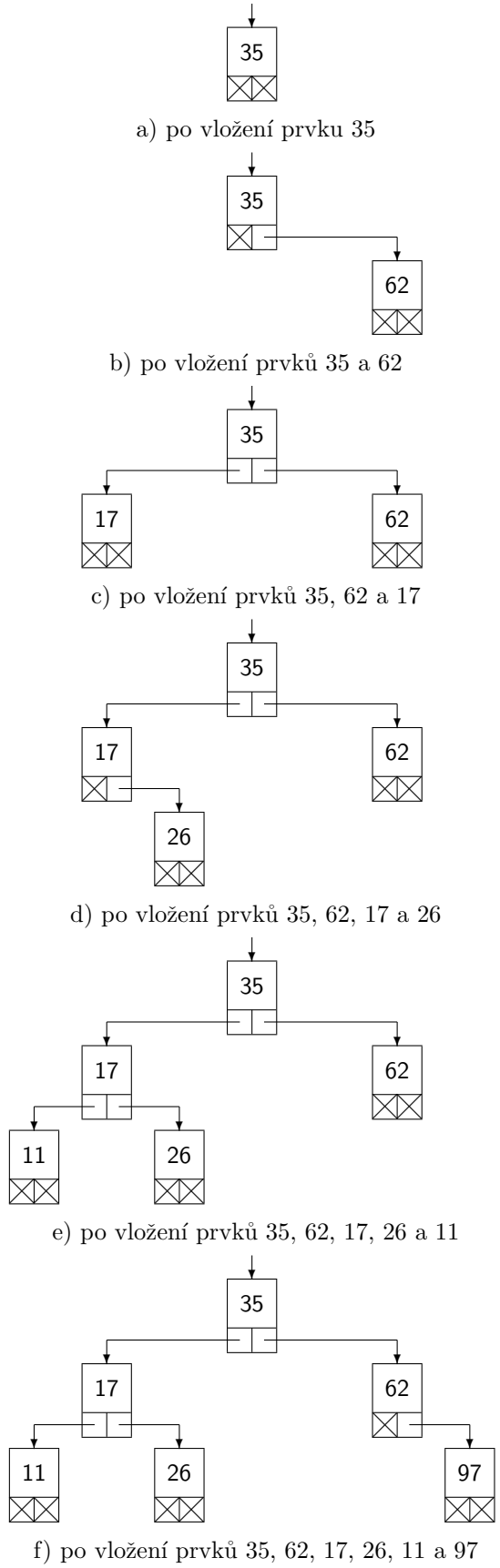
1.1.1 Vytvoření prázdného stromu

Strom je reprezentován provázanými uzly, na jeho kořen vede speciální odkaz. Při vzniku stromu stačí tento odkaz nastavit na prázdný odkaz – na `None`.

1.1.2 Vyhledání prvku

Při vyhledávání prvku ve stromě, na ketrý je nám dán odkaz (p) postupujeme takto:

- je-li odkaz prázdný (None), hledání končí – prvek nebyl nalezen;
- je-li prvek v odkazovaném uzlu roven hledanému, hledání končí – prvek byl nalezen v tomto uzlu;
- je-li hledaný prvek menší než prvek v odkazovaném uzlu, hledání pokračuje v levém podstromě;
- je-li hledaný prvek větší než prvek v odkazovaném uzlu, hledání pokračuje v pravém podstromě.



a) po vložení prvku 35

b) po vložení prvků 35 a 62

c) po vložení prvků 35, 62 a 17

d) po vložení prvků 35, 62, 17 a 26

e) po vložení prvků 35, 62, 17, 26 a 11

f) po vložení prvků 35, 62, 17, 26, 11 a 97

Obrázek 4: Postupné vkládání prvků do binárního vyhledávacího stromu

1.1.3 Přidání prvku

Uvažujeme práci s množinou.

- máme-li odkaz na prázdný strom, vytvoříme nový uzel a naplníme jej (oba odkazy na None, data na vkládaný prvek); konec
- srovnáme vkládaný prvek s prvkem v aktuálním uzlu:
  - jsou-li oba prvky stejné, prvek již v množině je a není třeba jej znovu vkládat
  - je-li vkládaný prvek menší, vložíme jej do levého podstromu aktuálního prvku
  - je-li vkládaný prvek větší, vložíme jej do pravého podstromu aktuálního prvku

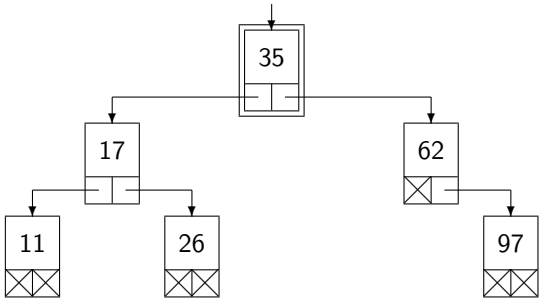
Příklad: Mějme čísla 35, 62, 17, 26, 11 a 97. Tato čísla jsou postupně vkládána do binárního vyhledávacího stromu – viz obr. 4.

1.1.4 Odebrání prvku

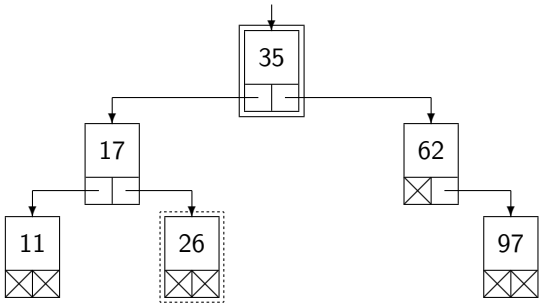
Při odebírání prvku postupujeme do jisté míry podobně jako při hledání či vkládání:

- Je-li (pod)strom prázdný, prvek ve stromě nebyl, a tak není co vypouštět. Konec.
- Je-li aktuální uzel tím, který se má vypouštět (prvek v něm se shoduje s tím, co se má vypustit), pak záleží na tom, kolik potomků daný uzel má:
  - 0) Uzel může být odstraněn
  - 1) Místo odkazu na uzel stačí nastavit odkaz na jeho potomka (a tak uzel vypustit)
  - 2) Aby bylo možné zachovat pořadí uzlů a přitom strom moc nepřerovnávat, je třeba prvek v uzlu nahradit buď bezprostředně větším prvkem nebo bezprostředně menším prvkem. Takový prvek najdeme v nejlevějším uzlu pravého podstromu či v nepravějším uzlu levého podstromu. Takový uzel má určitě nejvýše jednoho potomka, a tak půjde ze stromu snadno odstranit.

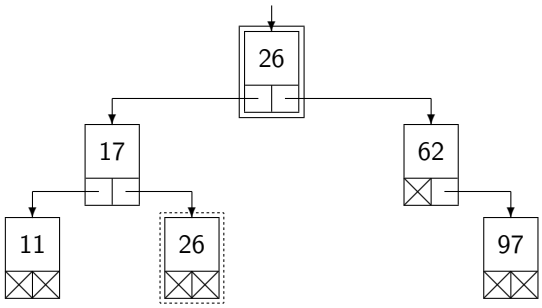
Příklad: Ze stromu vytvořeného v předchozím příkladu odebereme prvek 35 (má oba potomky). Prvek nalezneme snadno (je v kořeni). Abychom jej mohli vypustit, potřebujeme najít uzel s náhradní hodnotou. K tomuto účelu se hodí bezprotředně větší nebo bezprostředně menší hodnota. Ty nalezneme buď v nejlevějším uzlu pravého podstromu (62) nebo nejpravějším uzlu levého podstromu (26). Na obr. 5 je rozepsána varianta „nejpravější uzel levého podstromu“. Druhou variantu ponecháme na vlastní práci čtenáře.



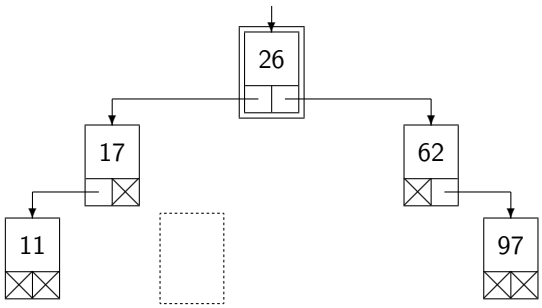
a) původní strom s vyznačeným prvkem k vypuštění



b) původní strom s vyznačeným prvkem k vypuštění a nalezeným uzlem s náhradní hodnotou (nejpravější v levém podstromě)



c) nahrazení vypouštěné hodnoty náhradní hodnotou



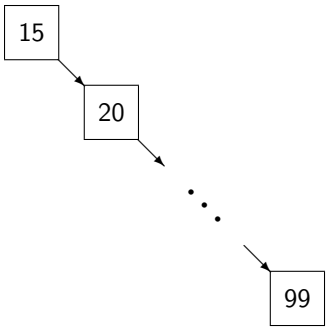
d) výsledný strom po vypuštění uzlu s náhradní hodnotou

Obrázek 5: Vypuštění uzlu se dvěma potomky (35)

2 Vyvažování: AVL strom

Binární vyhledávací strom může nabýt podoby, která pro většinu operací není moc vhodná – viz obr. 6. Takový strom odpovídá definici, ale operace nad ním jsou velmi pracné (časově náročné). Chceme-li se takové pdoby stromu vyvarovat, je třeba strom průběžně upravovat.

Za plně vyvážený strom budeme považovat takový, v němž pro každý z jeho uzlů platí, že počet uzlů v levém



Obrázek 6: Degradovaný binární vyhledávací strom

podstromě se neliší od počtu uzlů v pravém podstromě nejvýše o 1. Udržovat strom v takto přísně definované podobě je však velmi pracné. Až moc pracné pro stromy, které se často mění – do nichž se hodně přidávají nebo z nich odebírají prvky (uzly).

Budeme-li za ještě vážený považovat strom, kde pro každý jeho uzel plat, že výška levého a pravého podstromu se liší nejvýše o 1, získáme více volnosti, což umožní jednodušší a méně časté vyvažování. Získáváme tak variantu binárního vyhledávacího stromu, která má všechny operace s jednm prvkem realizovatelné nejhůře v $O(\log n)$ a přitom udržet celkovou pracnost v přijatelných mezích.

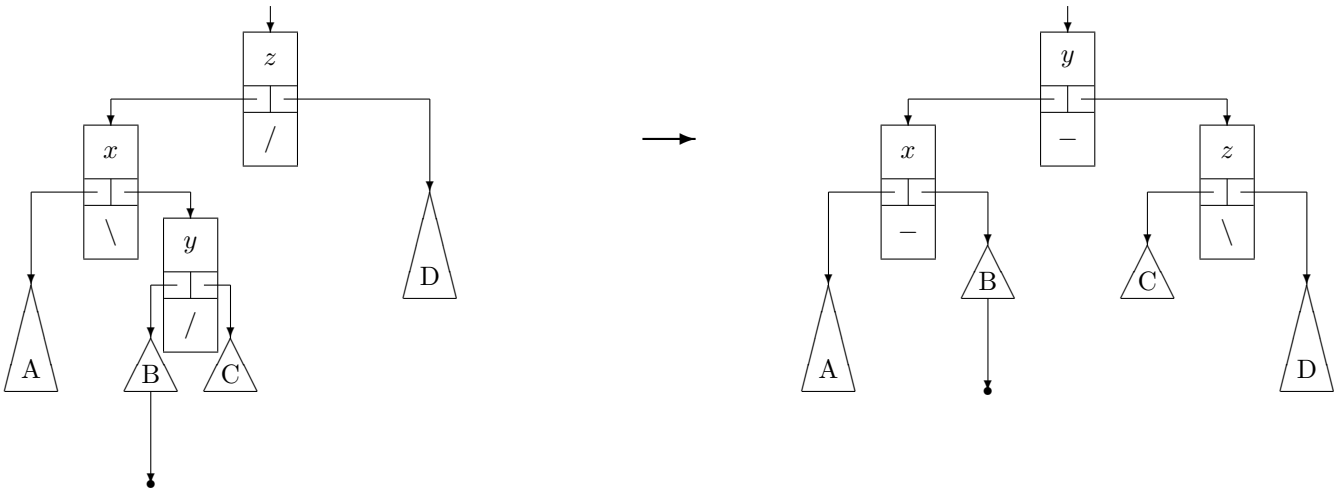
Jak ale takový strom implementovat? Abychom nemuseli neustále přepočítávat výšku podstromů, budeme si v každém uzlu pamatovat, jaký je v daném uzlu stav vyváženosti. Z definice se výška obou postromů nesmí lišit o více než 1, a tak bychom měli vystačit se třemi stavy:

1. levý podstrom je hlubší (/);
2. oba podstromy jsou stejně hluboké (−);
3. pravý podstrom je hlubší (\).

Jak budeme s „ukazovátkem“ pracovat? Při vzniku uzlu (bývá vkládán jako list) jsou oba ukazatelé zpravidla nastaveny na None. Oba podstromy jsou pak stejně hluboké, čemuž by mělo odpovídat i nastvení „ukazovátka“ na −.

Abychom udrželi ukazovátko aktuální, potřebujeme vědět, zda při vkládání do podstromu nedošlo ke změně jeho hloubky. Je proto nutné, aby vkládání do podstromu tuto informaci poskytovalo. Dojde-li ke změně hloubky podstromu, v mnoha případech stačí upravit výšku podstromu:

- byl-li aktuální uzel vyvážen (−) a došlo-li k zvětšení hloubky levého podstromu, stačí upravit ukazovátko na / a ohlásit změnu hloubky podstromu;
- byl-li aktuální uzel vyvážen (−) a došlo-li k zvětšení hloubky pravého podstromu, stačí upravit ukazovátko na \ a ohlásit změnu hloubky podstromu;
- měl-li aktuální uzel vyšší hloubku pravého podstromu (\) a došlo-li k zvětšení hloubky levého podstromu, stačí upravit ukazovátko na vyváženo (−) a ohlásit, že ke změně hloubky podstromu nedošlo;



Obrázek 7: LR-rotace

- měl-li aktuální uzel vyšší hloubku levého podstromu (/) a došlo-li k zvětšení hloubky levého podstromu, stačí upravit ukazovátko na vyváženo (−) a ohlásit, že ke změně hloubky podstromu nedošlo;

Zbývající dva případy se váží k situaci, kdy se zvětšila hloubka podstromu, který již hlubším byl – a tedy kdy došlo k porušení pravidla vyváženosti stromu v daném uzlu. To je třeba neprodleně řešit – je třeba strom vyvážit. K tomu jsou určeny čtyři operace (vždy ve dvojici symetrické) souhrnně označované jako *rotace* (označené LL, LR, RL a RR). To, která z rotací má být užita, je určeno stavem ukazovátek v uzlu, kde došlo k porušení vyrovnanosti a v uzlu, který je jeho potomkem na hlubší straně:

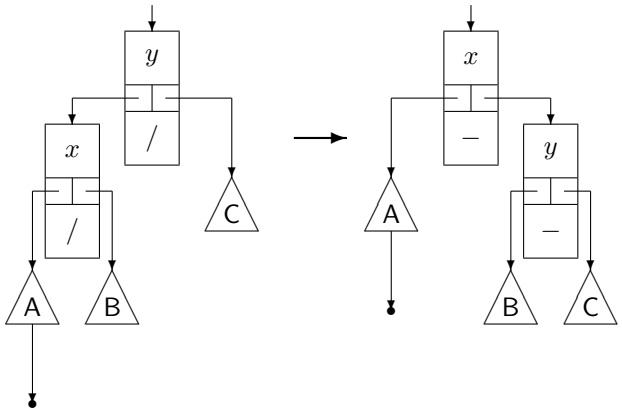
1. je-li těžší levý podstrom a jeho levý podstrom, užijeme LL-rotaci,
2. je-li těžší levý podstrom a jeho pravý podstrom, užijeme LR-rotaci,
3. je-li těžší pravý podstrom a jeho levý podstrom, užijeme RL-rotaci,
4. je-li těžší pravý podstrom a jeho pravý podstrom, užijeme RR-rotaci,

Rotace LL a RR, resp LR a RL, jsou navzájem zrcadlovými obrazy jedna druhé. Stačí si tak ukázat, jak funguje zástupce každé z dvojic.

2.1 LL-rotace

LL-rotace převěšuje tři podstromy mezi dvěma uzly a mění i vzájemnou polohu těchto dvou uzlů. Výsledkem je snížení celkové výšky daného podstromu o 1 a vyrovnání balance v obou zmíněných uzlech. Všechny tři zmíněné podstromy samy o sobě zůstávají beze změn. Schema LL-rotace je patrené z obr. 8.

Zde nabádáme čtenáře, aby si odvodil schema RR-rotace, která je zrcadlovým obrazem LL-rotace. LL a RR rotace bývají též označovány jako *jednoduché rotace*.



Obrázek 8: LL-rotace

2.2 LR-rotace

Přidávat lze i do „vnitřního podstromu“ – tedy třeba do pravého podstromu levého potomka. Nechť tento podstrom s kořenem y má dvě části – B a C, jako je tomu v levé části obr. 7. Nechť je přidán uzel k podstromu B takový, že mění jeho výšku („bambulka“ pod podstromem B).

Na laskavém čtenáři je ponecháno, aby se pokusil o LR transfomaci s tím, že nový uzel byl přidán do podstromu C (a nikoliv B jako na obrázku), resp. kdy nově přidaným uzlem byl sám uzel y. Následně je vhodné si odvodit i odpovídající varianty RL rotace.

3 Implementační tipy (Python)

V některých případech je třeba vracet z podprogramu více než jednu hodnotu. V takovém případě může funkce vracet `list` nebo `tuple` obsahující potřebné hodnoty ve vhodném pořadí. To se například týká vypouštění ze všech zmíněných variant binárních stromů (při vypouštění je třeba najít vhodnou náhradní hodnotu a případně i modifikovat odkaz) i vkládání do AVL stromů (kdy je třeba vracet jak modifikovaný podstrom, tak informaci o změně

výšky podstromu).