(a,b)-stromy

Zápočtová práce z Programování I pro pokročilé

Jiří Škrobánek[1]

21. listopadu 2018, Ostrava

Abstract

This documentation describes the entire functionality of (a,b)-trees implementation in Python 3 by Jiří Škrobánek. Aside from listing all methods, principles of (a,b)-trees are explained and complexity of used algorithms is briefly analysed.

Obsah

1 Definice

(a.b)-strom je strom. Musí platit $(a, b) \in \mathbb{N}^2, 2 \leq a, 2a - 1 \leq b$. (a,b)-strom je buďto prázdný, nebo mají všechny vnitřní vrcholy nejméně a synů a nejvýše b synů. Výjimkou je kořen, jenž musí mít mezi 2 a b syny. Všechny listy leží v jedné hladiny. Vnějším vrcholům jsou přiřazeny unikátní klíče (prvky lineárně uspořádané množiny). Vnitřním vrcholům je přiřazen maximální klíč z jeho synů. Ve vnitřním vrcholu jsou uloženy klíče synů ve vzestupném pořadí. Pro každý podstrom platí, že mimo podstrom neexistují vnější vrcholy, které mají nižší klíč než maximální klíč v podstromu a zároveň vyšší klíč než minimální v podstromu.
V tomto stromě se dá vyhledat vnější vrchol dle klíče v logaritmickém čase vzhledem k počtu vnějších vrcholů. Přidávání a odebírání listů rovněž funguje v logaritmickém čase.
Speciálním případem (a,b)-stromů pro $2a - 1 = b$ jsou B-stromy, mimo jiné oblíbený 2-3-strom.

[1]Matematicko-fyzikální fakulta Univerzity Karlovy, `jiri@skrobanek.cz`

2 Metody

Knihovna obsahuje třídy `InternalNode` a `ExternalNode` pro vnitřní a vnější vrcholy a třídu `ABTree`, která má metody popsané níže.
Počet vnějších vrcholů ve stromě značíme E.

```
__init__(a: int,b: int)
```

Vytvoří (a,b)-strom pro konkrétní hodnoty a, b.
Výjimky: `ValueError`: Neplatná volba parametrů.
Složitost: $\mathcal{O}(1)$

```
insert(self, key: int, value=None)
```

Vloží do stromu uspořádanou dvojici (key, value). Value může být libovolného typu nebo `None`.
Výjimky: `ValueError`: Strom již klíč obsahuje.
Složitost: $\Theta(\log(E))$

```
delete(self, key: int)
```

Vymaže ze stromu vnější vrchol s klíčem `key`.
Výjimky: `ValueError`: Strom klíč neobsahuje.
Složitost: $\Theta(\log(E))$

```
contains(self, key: int)
```

Vrací: `bool` Zda strom obsahuje daný klíč.
Složitost: $\Theta(\log(E))$

```
find(self, key: int)
```

Vrací: Hodnotu ve vnějším vrcholu s daným klíčem.
Výjimky: `ValueError`: Strom klíč neobsahuje.
Složitost: $\Theta(\log(E))$

```
item_count(self)
```

Vrací: `int` Počet vnějších vrcholů ve stromě.
Složitost: $\mathcal{O}(1)$

```
find_leq(self, key: int)
```

Vrací: `(k, value)` Uspořádaná dvojice klíče a hodnoty z vnějšího vrcholu, který má maximální klíč v množině vrcholů s klíčem v intervalu $(-\infty, key\rangle$
Složitost: $\mathcal{O}(\log E)$

```
find_lesser(self, key: int)
```

Vrací: `(k, value)` Uspořádaná dvojice klíče a hodnoty z vnějšího vrcholu, který má maximální klíč v množině vrcholů s klíčem v intervalu $(-\infty, key)$
Složitost: $\mathcal{O}(\log E)$

```
find_geq(self, key: int)
```

Vrací: `(k, value)` Uspořádaná dvojice klíče a hodnoty z vnějšího vrcholu, který má minimální klíč v množině vrcholů s klíčem v intervalu $\langle key, \infty)$
Složitost: $\mathcal{O}(\log E)$

```
find_greater(self, key: int)
```

Vrací: `(k, value)` Uspořádaná dvojice klíče a hodnoty z vnějšího vrcholu, který má minimální klíč v množině vrcholů s klíčem v intervalu (key, ∞)
Složitost: $\mathcal{O}(\log E)$

3 Příkladové použití

Na obrázku 1 vidíme použití stromu. Na vyhledání pomocí klíče můžeme použít funkci `__get_item__` a na zjištění počtu záznamů ve stromě `__len__`.

```
import abtree

tree = abtree.ABTree(2, 3)  # Create an empty tree, a = 2, b = 3

# Fill tree with entries:
tree.insert(10, 7), tree.insert(20, 2), tree.insert(0, 3)
tree.insert(30, 5), tree.insert(40, 1), tree.insert(50, 4)

# Remove an entry from the tree:
tree.delete(30)

print("Value at 10: " + str(tree[10]))
print("\nElement greater than 20: " + str(tree.find_greater(20)))
print("\nAmount of entries in the tree: " + str(len(tree)))
```

Obrázek 1: Příkladový program

Na obrázku 2 vidíme výstup získaný spuštěním příkladového programu.

4 Stručný popis algoritmů

4.1 Vkládání

Vkládáme `key, value`. Pokud je strom prázdný vytvoříme kořen, který bude mít nově vkládaný záznam jako jednoho syna a vrchol s klíčem ∞ jako druhého syna. Začneme u kořene. Z vrcholu přejdeme na jeho syna, který má nejbližší vyšší klíč než `key`.
Po vložení zavoláme algoritmus vyvažování na otce nově vzniklého vnějšího vrcholu.

4.2 Hledání

Pokud hledáme podle klíče `key`, začneme v kořeni a sestupujeme na syny s nejbližším vyšším klíčem, nebo případně klíčem rovným `key`. Pokud nedojdeme do vnějšího vrcholu s hledaným klíčem, strom klíč neobsahuje.

4.3 Mazání

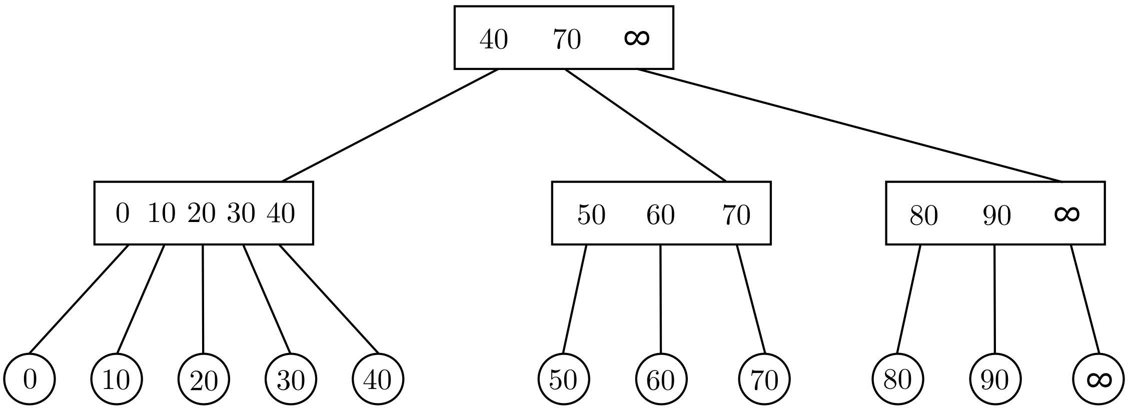
Pokud je třeba vrchol vymazat, nejdříve ho je třeba najít ve stromě a potom odstranit. Pokud byl navíc maximálním klíčem v některých vnitřních vrcholech, je třeba změnit v jeho předcích klíče na druhý nejvyšší vrchol v otci mazaného vrcholu (Takový musí existovat.)
Může být porušeno vyvážení stromu, proto je třeba zavolat vyvažování na bývalého otce smazaného vrcholu.

```
Value at 10: 7

Element greater than 20: (40, 1)

Amount of entries in the tree: 5
```

Obrázek 2: Výstup příkladového programu



Obrázek 3: Příkladový (3,5)-strom

4.4 Vyvažování

Pokud jsme přidávali nebo mazali od posledního spuštění vyvažování pouze jeden vrchol, povolený počet synů ve vrcholech může být porušen o nejvýše 1.
Pokud ve stromě zůstal kořen a jeden vrchol (∞), strom se stane prázdným.
Kontrolu stromu vždy začínáme v nějakém vrcholu. A může se vyvíjet několika způsoby:

- Pokud je kontrolovaný vrchol kořen, který má jediného syna, odstraníme ho z grafu a jeho syn se stane novým kořenem. Ukončíme vyvažování.
- Pokud je kontrolovaný vrchol kořen a má více než b synů, rozdělíme ho na dva a vytvoříme jim nového otce, který bude kořenem stromu. Ukončíme vyvažování.
- Pokud má kontrolovaný vrchol více než b synů: Rozdělíme jej na dva, oba vrcholy získají nejméně a synů, rozdělených podle velikosti. Tím mohl otec získat příliš mnoho synů, kontrolu pokračujeme na něm.
- Pokud má kontrolovaný vrchol méně než a synů: Zkontrolujeme, zda se nemůže levý nebo pravý rozdělit o syny s tímto vrcholem, aby měly oba alespoň a synů. Pokud toto nejde, sloučením s jedním z nich vznikne vrchol s povoleným počtem synů. Poté je potřeba pokračovat s kontrolou v otci, protože ztratil jednoho syna.
- Pokud je vše s vrcholem v pořádku, vyvažování ukončíme.

Výsledkem vyvažování je platný (a,b)-strom.

5 Analýza složitosti

5.1 Minimální vyváženost stromu

Pokud má strom e vnějších vrcholů, strom má hloubku nejvýše $\lceil \log_a e \rceil + 1$.
Z definice musejí mít všechny vnitřní vrcholy nejméně a synů, a proto počet vrcholů hloubky h ve stromě je nejméně a^h.

5.2 Paměťová složitost

Strom s parametry (a, b) a e záznamy zabírá v paměti $\Theta(e)$. Neuvažujeme, kolik zabírají samotné údaje, které do stromu ukládáme, pouze strukturu stromu a klíče. Strom má nejvýše hloubku $\lceil \log_a e \rceil + 1$. Jeden vnitřní vrchol v sobě obsahuje klíče a ukazatele na nejvýše b dalších vrcholů.
Můžeme tedy provést horní odhad tak, že všechny hladiny budou plné a bude jich teoretický maximální počet. V tomto případě je počet vnějších vrcholů roven a. Další úroveň obsahuje ve vrcholech klíče a ukazatele na externí vrcholy, ale všechny další obsahují nejméně a-krát méně klíčů, je jich a-krát méně. Součet takovéto geometrické řady je vždy pouze konstantním násobkem prvního členu, tedy e.

5.3 Rychlost vyhledávání

Vyhledávání se použije nejen v případě, že je potřeba získat z pole záznam, ale také při vkládání a odstraňování, je tedy nutné, aby probíhalo rychle.
Jak plyne z algoritmu, začíná se ve stromě a sestupuje se až na spodní hladinu. V každém prošlém vrcholu se přitom porovná až b klíčů. Pokud je hledání nakonec neúspěšné, proces se liší až posledním krokem.
Dohromady to dává $\Theta(b \log_a e)$ operací ve stromě s e záznamy.

5.4 Rychlost vyvažování

Vyvažování spouštíme na nějakém konkrétním vrcholu, pouze pro tento vrchol hrozí porušení počtu synů. Provádí se de facto dvě operace:
Slučování Pokud se uskuteční sloučení vrcholu, vrchol vzniklý po sloučení bude mít počet synů v pořádku. Jen jeho otce bude třeba preventivně zkusit vyvážit (pokud existuje). Nově vzniklý vrchol už je ale vyvážený a nebude se k němu potřeba vracet.
Hledání vrcholů ke sloučení i vyrábění nového vrcholu je práce s $\mathcal{O}(b)$ klíči.
Rozdělování Vrcholy vzniklé rozdělením mají správný počet synů, vyvažování pokračuje otcem těchto vrcholů.
Protože máme již určeno jaká je maximální hloubka stromu a vyvažuje se maximálně 1 vrchol v každé hloubce, můžeme říci, že během vyvažování proběhne $\mathcal{O}(e)$ operací.

Seznam obrázků

Reference

[1] KNUTH, Donald Ervin. The Art of Computer Programming. Upper Saddle River, NJ: Addison-Wesley, 2011. ISBN 978-0321751041.