(a,b)-stromy

Zápočtová práce z Programování I pro pokročilé

Jiří Škrobánek[1]

28. října 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 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 $2a - 1 = b$ jsou B-stromy, mimo jiné oblíbený 2-3-strom.

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)
```

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

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 Popis algoritmů

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.
Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

4 Analýza složitosti

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.
Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

Seznam obrázků

Seznam tabulek

Seznam příloh

Reference

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